МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)

Е.И. КОНОВАЛОВА, Л.В. ЯБЛОКОВА

ЧИСЛЕННЫЕ МЕТОДЫ ЛИНЕЙНОЙ АЛГЕБРЫ

Рекомендовано редакционно-издательским советом федерального государственного автономного образовательного учреждения высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева» в качестве учебного пособия для обучающихся по основной образовательной программе высшего образования по направлению подготовки 01.03.02 Прикладная математика и информатика

САМАРА
Издательство Самарского университета
2022

УДК 512.64(075)
ББК 22.193я7
К647

Рецензенты: д-р физ.-мат. наук, проф. А. И. Жданов,
д-р техн. наук, проф. В. В. Любимов

Коновалова, Елена Игоревна

К647 **Численные методы линейной алгебры**: учебное пособие / *Е.И. Коновалова, Л.В. Яблокова*. – Самара: Издательство Самарского университета, 2022. – 152 с.: ил.

ISBN 978-5-7883-1845-5

Учебное пособие организовано в виде набора глав, посвящённых определённым разделам численных методов. Каждая глава сопровождается примерами, упражнениями и лабораторными работами, направленными на повышение качества усвоения материала.

Предназначено для обучающихся по основным образовательным программам высшего образования по направлению подготовки 01.03.02 Прикладная математика и информатика и специальности 10.05.03 Информационная безопасность автоматизированных систем.

Подготовлено на кафедре прикладных математики и физики.

УДК 512.64(075)
ББК 22.193я7

ISBN 978-5-7883-1845-5

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В настоящее время численные методы являются универсальным математическим средством решения многих научно-технических проблем. Численные методы – это раздел вычислительной математики, изучающий способы решения типовых математических задач, которые либо не имеют точного аналитического решения, либо трудно решаются традиционными методами. Численные методы лежат в основе таких современных направлений как машинное обучение и нейронные сети. Классическим примером является задача линейной регрессии в машинном обучении, кото-рая сводится к решению переопределенной системы линейных уравнений. Авторы полагают, что для будущих специалистов совершенно необходимо понимать алгоритмы, которые реализова-ны во всех современных библиотеках, используемых в системах разработки на языках программирования Python, C++, FORTRAN и др.

При подготовке теоретического курса авторы особенно руководствовались классическими учебниками Амосова А.А., Дубинского Ю.А. и Копченовой Н.В. «Вычислительные методы» [1] и Демидовича Б.П. и Марона И.А. «Основы вычислительной математики» [11].

Учебное пособие написано на основе курса лекций для студентов, обучающихся по направлению подготовки 01.03.02 Прикладная математика и информатика и специальности 10.05.03 Информационная безопасность автоматизированных систем. Первая глава посвящена погрешностям вычислений, вторая – вычислению норм векторов и матриц, третья – решению систем линейных уравнений, в четвертой главе приводятся индивидуальные домаш-ние задания. Данное пособие является первой частью курса чис-ленных методов. Вторая часть курса представлена в учебном пособии «Численные методы математического анализа».

Оба пособия вместе представляют собой полный курс лекций по численным методам, читающийся в институте информатики и кибернетики Самарского университета и соответствуют учебной программе по этому курсу.

Авторы выражают глубокую благодарность директору института информатики и кибернетики Самарского университета Куприянову Александру Викторовичу за помощь в подготовке учебного пособия.

1 ВВОДНАЯ ГЛАВА

Процесс формирования решения вычислительных задач для конкретной предметной области связан с определённой стадийностью. На первом этапе необходимо провести постановку задачи, сформулировав её с точки зрения особенностей выбранного прикладного направления. Второй этап – математическая постановка, когда происходит ретрансляция проблемы на математический язык, средствами которого математическая модель могла бы корректно описывать исследуемые процессы или явления. Затем, оперируя методами абстрактной математики, решение описывается в общем виде, без использования конечных чисел, с применением формул, функций и абстрактных величин. После этого, с помощью алгоритмизации и программных технологий, решение задачи пытаются свести к выполнению конечного количества арифметических операций и упорядочить производимые действия в виде точного, воспроизводимого метода с программной реализацией на наиболее подходящем для этого языке программирования. В течение всего решения задачи этапам моделирования, алгоритмизации и программирования необходимо уделять особое внимание, потому что на каждой из этих стадий формируется концептуальная схема, включающая терминологический, понятийный аппараты, методы и средства с помощью которых будут проводиться расчёты и анализироваться результаты.

Но что значит решить математическую задачу? Только в исключительных случаях удается найти решение в явном виде, например в виде ряда. Иногда утверждение «задача решена» означает, что доказано существование и единственность решения. Ясно, что этого недостаточно для практических приложений. Необходимо еще изучить качественное поведение решения и найти те или иные количественные характеристики. Именно на этом этапе

требуется привлечение численных методов. Чтобы реализовать численный метод, необходимо составить программу для ЭВМ или воспользоваться готовой программой. После отладки программы наступает этап проведения вычислений и анализа результатов. Полученные результаты изучаются с точки зрения их соответствия исследуемому явлению и, при необходимости, вносятся исправления в численный метод и уточняется математическая модель.

В общих чертах схема вычислительного эксперимента такова: основу составляет триада: *модель – метод (алгоритм) – программа.*

1.1 Требования к вычислительным методам

Одной и той же математической задаче можно поставить в соответствие множество различных дискретных моделей. Однако далеко не все из них пригодны для практической реализации.

Можно выделить две группы требований к численным методам. Первая группа связана с адекватностью дискретной модели исходной математической задачи, и вторая группа – с реализуемостью численного метода на ЭВМ.

К первой группе относятся такие требования, как сходимость численного метода, выполнение дискретных аналогов законов сохранения, качественно правильное поведение решения дискретной задачи.

Поясним эти требования. Предположим, что дискретная модель математической задачи представляет собой систему большого, но конечного числа алгебраических уравнений. Обычно, чем точнее мы хотим получить решение, тем больше уравнений приходится брать.

Говорят, что численный метод сходится, если при неограниченном увеличении числа уравнений решение дискретной задачи стремится к решению исходной задачи.

Поскольку реальная ЭВМ может оперировать лишь с конечным числом уравнений, на практике сходимость, как правило, не достигается. Поэтому важно уметь оценивать погрешность метода в зависимости от числа уравнений, составляющих дискретную модель. По этой же причине стараются строить дискретную модель таким образом, чтобы она правильно отражала качественное поведение решения исходной задачи даже при сравнительно небольшом числе уравнений.

Сходимость численного метода тесно связана с его корректностью. Предположим, что исходная математическая задача поставлена корректно, т. е. ее решение существует, единственно и непрерывно зависит от входных данных. Тогда дискретная модель этой задачи должна быть построена таким образом, чтобы свойство корректности сохранилось. Таким образом, в понятие корректности численного метода включаются свойства однозначной разрешимости соответствующей системы уравнений и ее устойчивости по входным данным.

Под *устойчивостью* понимается непрерывная зависимость решения от входных данных, равномерная относительно числа уравнений, составляющих дискретную модель.

Решение задачи y^* называется устойчивым по исходным данным x^*, если оно зависит от исходных данных непрерывным образом. Это означает, что малому изменению исходных данных соответствует малое изменение решения. Строго говоря, для любого $\varepsilon > 0$ существует $\delta=\delta(\varepsilon) > 0$ такое, что всякому исходному данному x^* , удовлетворяющему условию $|x - x^*| < \delta$, соответствует приближенное решение y^*, для которого $|y - y^*| < \varepsilon$.

Говорят, что задача поставлена *корректно*, если выполнены следующие три условия:

1. Решение существует при любых допустимых исходных данных.

2. Это решение единственно.

3. Это решение устойчиво по отношению к малым изменениям исходных данных.

Если хотя бы одно из этих условий не выполнено, задача называется *некорректной*.

Вторая группа требований, предъявляемых к численным методам, связана с возможностью реализации данной дискретной модели на данной ЭВМ, т. е. с возможностью получить на ЭВМ решение соответствующей системы алгебраических уравнений за приемлемое время. Основным препятствием для реализации корректно поставленного алгоритма является ограниченный объем оперативной памяти ЭВМ и ограниченные ресурсы времени счета. Реальные вычислительные алгоритмы должны учитывать эти обстоятельства, т. е. они должны быть экономичными как по числу арифметических действий, так и по требуемому объему памяти.

1.2 Источники и классификация погрешности

При численном решении математических и прикладных задач, почти неизбежно появление на том или ином этапе их решения погрешностей возникающих по следующим причинам:

1) математическое описание задачи является неточным, в частности неточно заданы исходные данные описания. Погрешности в решении, обусловленные моделированием и исходными данными, называются *неустранимыми*;

2) применяемый для решения метод не является точным. Всякий численный метод воспроизводит исходную математическую модель приближенно, при этом неизбежно возникает *погрешность метода вычислений*;

3) при вводе данных, при выполнении арифметических операций и при выводе данных производятся округления. Так возникает *погрешность округления*, которая может накапливаться в

ходе вычислений (опасный процесс, способный обесценить результат вычислений!).

Все три типа таких погрешностей в сумме дают полную погрешность результата решения задачи.

Алгоритм называется *устойчивым***,** если в процессе его работы вычислительные погрешности возрастают незначительно, и *неустойчивым* — в противоположном случае.

Итак, следует различать погрешности модели, метода и вычислительную. Какая же из этих трех погрешностей является преобладающей? Ответ здесь неоднозначен. Видимо, типичной является ситуация, возникающая при решении задач математической физики, когда погрешность модели значительно превышает погрешность метода, а погрешностью округления в случае устойчивых алгоритмов можно пренебречь по сравнению с погрешностью метода. С другой стороны, при решении, например, систем обыкновенных дифференциальных уравнений возможно применение столь точных методов, что их погрешность будет сравнима с погрешностью округления.

В общем случае нужно стремиться, чтобы все указанные погрешности имели один и тот же порядок.

1.3 Приближенные числа, их абсолютные и относительные погрешности

Расчеты, как правило, производятся с приближенными значениями величин, приближенными числами. Разумная оценка погрешности при вычислениях позволяет указать оптимальное количество знаков, которые следует сохранять при расчетах, а также в окончательном результате.

Рассмотрим несколько возможных подходов к учету погрешностей действий.

Пусть A и a – два «близких числа»; A – точное, a – приближенное.

Величина $\Delta a = |A - a|$ называется *абсолютной погрешностью* приближенного числа. Под оценкой погрешности приближенного числа a понимают установление неравенства вида $|A - a| \leqslant \Delta a$. Число Δa также называют *предельной абсолютной погрешностью* приближенного числа.

Абсолютные погрешности записывают не более чем с двумя-тремя значащими цифрами (значащими цифрами числа в его позиционной записи называются все его цифры, начинающиеся с первой ненулевой слева; например, в числе 0,01030 имеется четыре значащих цифры). В приближенном числе не следует сохранять те разряды, которые подвергаются округлению в его абсолютной погрешности.

Пример 1. Длина и ширина комнаты, измеренные с точностью до 1 см, равны a = 5,43 м и b = 3,82 м. Оценить погрешность в определении площади комнаты $S = ab$ = 20,7426 м^2.

Решение. По условию задачи Δa = 0,01 м, Δb = 0,01 м.

Крайние возможные значения площади равны

$$(a + 0{,}01)(b + 0{,}01) = 20{,}8352 \text{ м}^2;$$

$$(a - 0{,}01)(b - 0{,}01) = 20{,}6502 \text{ м}^2;$$

сравнивая их с подсчитанным выше значением S, получаем оценку $|S - S_0| < 0{,}0926$, что дает возможность указать абсолютную погрешность числа S в виде ΔS = 0,0926 м^2. При этом приближенное значение площади можно записать в виде S = 20,74 м^2.

Итак, абсолютная погрешность оценивает точность измерений, но эта оценка неполна, поскольку не учитывает характерный размер изучаемого явления (объекта). Так, например, абсолютная погрешность в 1 см при измерении длины комнаты – вероятно,

вполне приемлемая точность, но при измерении роста человека эта же погрешность будет сочтена непозволительно грубой.

Более информативным показателем качества измерений является относительная погрешность.

Относительной погрешностью δa приближенного числа a называется отношение его абсолютной погрешности Δa к абсолютной величине числа a т. е. $\delta a = \frac{\Delta a}{|a|}$ ($a \neq 0$), иногда $\delta a = \frac{\Delta a}{|A|}$. Относительная погрешность обычно выражается в процентах, и ее принято записывать не более чем с двумя-тремя значащими цифрами (знаками).

Пример 2. Определить относительную погрешность числа S в примере 1.

Решение. $S = 20{,}7426$, $\Delta S = 0{,}0926$, поэтому

$$\delta S = \frac{0{,}0926}{20{,}7426} = 0{,}0045 = 0{,}45\% .$$

Во многих технических приложениях принято характеризовать точность приближенных чисел их относительной погрешностью.

Относительная погрешность является величиной безразмерной, т. е. не зависит от выбора системы единиц измерения, что позволяет сравнивать качество измерений разнородных величин. Бессмысленным является вопрос о том, что больше: 1 кг или 1 м, но сравнение качества измерений массы и длины в терминах относительной погрешности вполне допустимо. Измеряется δa в долях единицы или в процентах.

Пример 3. Согласно ныне действующим определениям международного Комитета по константам для науки и технологии входящая в закон всемирного тяготения гравитационная постоянная $G = (6{,}67259 \pm 0{,}00085)\,10^{-11}$ м^3·кг$^{-1}$·с$^{-2}$, а заряд электрона

$e=(1{,}60217733\pm 0{,}00000049)\cdot 10^{-19}$ Кл. Сравнить точность определения этих фундаментальных физических постоянных.

Решение. Для гравитационной постоянной предельная относительная погрешность

$$\delta G=\frac{0{,}00085}{6{,}67259}=1{,}27\cdot 10^{-4},$$

а для заряда электрона

$$\delta e=\frac{0{,}00000049}{1{,}60217733}=3{,}1\cdot 10^{-7}.$$

Таким образом, в последнем случае относительная погрешность оказывается на три порядка меньшей, т. е. заряд электрона определен существенно точнее , чем гравитационная постоянная.

С понятиями абсолютной и относительной погрешностей связаны понятия верных и значащих цифр.

Если абсолютная погрешность приближенного числа не превышает единицы последнего (самого правого) разряда его десятичной записи, то цифры числа называют верными (или точными).

По умолчанию десятичная запись приближенного числа должна содержать только верные цифры, и тогда по записи числа сразу можно узнать предельную абсолютную погрешность, с которой оно известно. Цифры, не являющиеся верными, называются сомнительными.

Пример 4. Даны приближенные числа $a=8{,}6$; $b=8{,}60$; $c=3200$; $d=3{,}2\cdot 10^{3}$. Указать предельную абсолютную погрешность для каждого числа.

Решение. Для числа a погрешность $\Delta a\leqslant 0{,}1$, для числа b $\Delta b\leqslant 0{,}01$, для числа c $\Delta c\leqslant 1$, для числа d $\Delta d\leqslant 0{,}1\cdot 10^{3}=100$.

Итак, числа a и b, c и d, равные с точки зрения «обычной» математики, существенно различны в вычислительной математике:

из абсолютной погрешности мы заключаем, что число b известно точнее, чем число a, а число c – точнее, чем d. Кроме того, нуль, стоящий справа в дробной части десятичного числа, важен, и им нельзя пренебрегать, если мы хотим составить верное суждение о точности числа.

Значащими цифрами приближенного числа называются все цифры его десятичной записи, кроме нулей, находящихся левее первой отличной от нуля цифры.

Пример 5. Сколько значащих цифр имеют числа 0,001307 и 6,0400?

Решение. Числа 0,001307 и 6,0400 имеют соответственно четыре и пять значащих цифр т. к. нули, находящиеся слева, значащими не являются, а нуль, записанный в конце десятичной дроби, всегда является значащей цифрой.

Количество верных знаков числа отсчитывается от первой значащей цифры числа до первой значащей цифры его абсолютной погрешности: например, число $S = 20{,}7426$ с абсолютной погрешностью $\Delta S = 0{,}0926$, имеет три верных знака (2, 0, 7); остальные знаки сомнительные.

Ориентировочно можно считать, что наличие только одного верного знака соответствует относительной погрешности порядка 10%, двух верных знаков – погрешности порядка 1% и т. д.

В математических таблицах все числа округлены до верных знаков, причем абсолютная погрешность не превосходит половины единицы последнего оставленного разряда. Например, если в таблице указано $e = 2{,}718$, то абсолютная погрешность не превосходит $0{,}5 \cdot 10^{-3}$.

В окончательных результатах вычислений обычно оставляют, кроме верных, один сомнительный знак.

В промежуточных результатах вычислений обычно сохраняют два-три сомнительных знака, чтобы не накапливать лишних погрешностей от округлений.

Пример 6. Округлить число $S = 20{,}7426$ в примере 1 до верных знаков.

Решение. Так как в числе S три верных знака, то естественно записать $S = 20{,}7$.

Однако при этом к абсолютной погрешности $\Delta S = 0{,}0926$ приходится добавить еще величину 0,0426, отброшенную при округлении. Новая абсолютная погрешность $\Delta S = 0{,}136$ заставляет считать сомнительным уже третий знак числа S, и, следовательно, число S приходится округлять до двух знаков: $S = 21$.

Этот пример показывает, что округление результатов расчета до верных знаков не всегда целесообразно.

Числа Δ_a и δ_a такие, что $\Delta_a \geqslant \Delta a$ и $\delta_a = \frac{\Delta_a}{|a|} \geqslant \delta a$ называются оценками (границами) абсолютной и относительной погрешностей соответственно (предельные погрешности). Для их расчета используется аналитический (классический) способ учета погрешностей действий, предполагающий точное оценивание погрешностей, основанное либо на правилах подсчета погрешностей арифметических действий, либо на параллельной работе с верхними и нижними границами исходных данных. Этот способ громоздок, учитывает крайние, наихудшие случаи взаимодействия погрешностей.

Существуют вероятностно-статистические законы, которые используются при больших однотипных вычислениях. Тоже достаточно сложно и вряд ли может быть рекомендовано при рядовых массовых вычислениях. Технический подход связан с именем известного русского кораблестроителя, математика и механика Алексея Николаевича Крылова.

Принцип А.Н. Крылова: приближенное число должно записываться так, чтобы в нем все значащие цифры, кроме последней, были верными и лишь последняя была бы сомнительна, и притом в среднем (в вероятностном смысле) не более чем на единицу.

Значащую цифру называют верной (в широком смысле), если абсолютная погрешность числа не превосходит единицы разряда, в которой стоит эта цифра (или половины единицы разряда – в этом случае термин – «верная в узком смысле»).

Чтобы результаты арифметических действий, совершаемых над приближенными числами, записанными в соответствии с принципом А.Н. Крылова, тоже соответствовали этому принципу нужно придерживаться следующих простых правил.

1. При сложении и вычитании приближенных чисел в результате следует сохранять столько десятичных данных, сколько их в приближенном данном с наименьшим количеством десятичных знаков;

2. При умножении и делении в результате нужно сохранять столько значащих цифр, сколько их имеет приближенное данное с наименьшим числом значащих цифр;

3. Результаты промежуточных данных должны иметь 1-2 запасных знаков, затем их отбрасывают.

Выдача числовых значений в ЭВМ, как правило, устроена таким образом, что нули в конце записи числа, даже если они верны, не сообщаются. Это означает, что если, например ЭВМ показывает результат 236,057 и в тоже время известно, что в этом результате верными должны быть 8 значащих цифр, то полученный ответ следует дополнить двумя нулями 236,05700.

При округлении числа мы заменяем его приближенным числом с меньшим количеством значащих цифр, в результате чего возникает погрешность округления.

Правила округления числа, т. е. его замены числом с меньшим количеством значащих цифр:

1. Если первая слева из отбрасываемых цифр больше либо равна 5, то последняя из сохраняемых цифр усиливается, т. е. увеличивается на единицу.

2. Если первая из отброшенных цифр меньше 5, то последняя из оставшихся цифр не усиливается, т. е. остается без изменения.

3. Если первая слева из отброшенных цифр равна 5 и за ней не следует отличных от нуля цифр, то последняя оставшаяся цифра усиливается, если она нечетная, и остается без изменения, если она четная (правило четной цифры).

Пример 7. Округлить числа 5,785 и 5,775 до сотых.

Решение. Округляя число 5,785 до сотых получаем 5,78. Усиление не делаем т. к. последняя сохраняемая цифра «8» – четная. Число 5,775 округляем до второго десятичного знака, имеем 5,78. Последняя сохраняемая цифра «7» увеличивается на единицу, т. к. «7» – нечетная.

Смысл правила 3 в том, что при многочисленных округлениях избыточные числа будут встречаться примерно с той же частотой, что и недостаточные, и произойдет частичная взаимная компенсация погрешностей округления; результат окажется более точным.

При использовании правил округления – абсолютная погрешность округления не превосходит половины единицы разряда, определяемого последней оставленной значащей цифрой.

Абсолютная погрешность алгебраической суммы нескольких приближенных чисел не превышает суммы абсолютных погрешностей этих чисел.

При сложении чисел различной абсолютной точности обычно поступают следующим образом:

1) выделяют число (или числа) наименьшей абсолютной точности (т. е. число, имеющее наибольшую абсолютную погрешность);
2) наиболее точные числа округляют так, чтобы сохранить в них на один знак больше, чем в выделенном числе (т. е. оставить один запасной знак);
3) производят сложение, учитывая все сохраненные знаки;
4) полученный результат округляют на один знак.

Пример 8. Сложить приближенные числа 0,1732; 17,45; 0,000333; 204,4; 7,25; 144,2; 0,0112; 0,634; 0,0771, считая в них все знаки верными, т. е. считая, что абсолютная погрешность каждого слагаемого не превосходит половины единицы младшего оставленного разряда.

Решение. Числа с наименьшей точностью: 204,4 и 144,2 – верны с точностью до 0,05. Поэтому можно считать, что абсолютная погрешность суммы составляет $2\Delta = 0{,}10$. Так как количество слагаемых невелико, то в расчетах сохраняем только один запасной знак, т. е. округляем слагаемые до 0,01: 0,1732≈0,17; 17,45≈17,45; 0,000333≈0,00; 7,25≈7,25; 0,0112≈0,01; 0,634≈0,63; 0,0771≈0,08. Складываем полученные числа с точностью до 0,01. Округляя результат до одного знака после запятой, получим окончательный ответ: 374,19≈374,2.

Пример 9. Оценить относительную погрешность суммы чисел в примере 8 и сравнить ее с относительными погрешностями слагаемых.

Решение. Относительная погрешность суммы S равна $\delta S = \frac{0{,}10}{374{,}19} = 0{,}0003$.

Относительные погрешности слагаемых составляют соответственно $\frac{0{,}05}{0{,}1732} = 0{,}29$, $\frac{0{,}05}{17{,}32} = 0{,}003$, $\frac{0{,}05}{204{,}4} = 0{,}0002$,

$\frac{0,05}{7,25}=0,007$, $\frac{0,05}{144,2}=0,0003$, $\frac{0,05}{0,0112}=4,46$, $\frac{0,05}{0,634}=0,08$, $\frac{0,05}{0,0771}=0,004$.

Предельная относительная погрешность суммы слагаемых одного знака заключена между наименьшей и наибольшей предельными относительными погрешностями слагаемых.

Относительная погрешность разности двух положительных чисел больше относительных погрешностей этих чисел. При вычитании близких чисел часто возникает положение, называемое потерей точности. Пусть $x>0$, $y>0$ и $a=x-y$, тогда если числа x и y мало отличаются друг от друга, то даже при малых погрешностях Δx , Δy величина относительной погрешности разности может оказаться значительной

$$\delta a=\frac{\Delta a}{|a|}=\frac{\Delta x+\Delta y}{|x-y|}.$$

Пример 10. Даны числа a = 1,137 и b =1,073 с абсолютными погрешностями $\Delta a=\Delta b=0,011$. Оценить погрешность их разности $c=a-b$.

Решение. c=0,064,

$$\Delta c=\Delta a+\Delta b=0,022 ,\ \delta c=\frac{0,022}{0,064}=0,34=34\% .$$

Таким образом, в результате нет ни одного верного знака, хотя сами числа имеют относительные погрешности $\delta a\approx\delta b\approx 1\%$.

При вычитании близких чисел может произойти большая потеря точности, чтобы не допустить этого необходимо их брать с достаточным числом запасных верных знаков.

Пример 11. Найти разность $a=\sqrt{6,27}-\sqrt{6,26}$ и оценить относительную погрешность результата.

Решение. $a_1=\sqrt{6{,}27}=2{,}504$, $\Delta a_1=0{,}0005$,

$a_2=\sqrt{6{,}26}=2{,}502$, $\Delta a_2=0{,}0005$.

Тогда $a=2{,}504-2{,}502=0{,}002$, $\Delta a=0{,}0005+0{,}0005=0{,}001$, $\delta a=\dfrac{0{,}001}{0{,}002}=0{,}5=50\%$.

Однако, изменив вычислительную схему, можно получить:

$$a=\sqrt{6{,}27}-\sqrt{6{,}26}=\frac{(\sqrt{6{,}27}-\sqrt{6{,}26})\cdot(\sqrt{6{,}27}+\sqrt{6{,}26})}{\sqrt{6{,}27}+\sqrt{6{,}26}}=$$

$$=\frac{6{,}27-6{,}26}{\sqrt{6{,}27}+\sqrt{6{,}26}}=\frac{0{,}01}{\sqrt{6{,}27}+\sqrt{6{,}26}}\approx\frac{0{,}01}{2{,}504+2{,}502}=0{,}002,$$

$$\delta a=\frac{\Delta a_1+\Delta a_2}{a_1+a_2}=\frac{0{,}001}{5{,}006}=0{,}0002=0{,}02\% .$$

Таким образом, получили лучший результат относительной погрешности.

При умножении и делении приближенных чисел складываются их относительные погрешности (а не абсолютные!); относительная погрешность выражения

$$r=\frac{a_1\cdot a_2\cdot\ldots\cdot a_m}{b_1\cdot b_2\ldots\cdot b_n}$$

оценивается величиной

$$\delta r=\delta a_1+\delta a_2+\ldots+\delta a_m+\delta b_1+\delta b_2+\ldots+\delta b_n .$$

При большом числе $m+n$ выгоднее пользоваться статистической оценкой, учитывающей частичную компенсацию погрешностей разных знаков: если все числа a_i и b_j (i = 1, 2, ..., m, j = 1, 2, ..., n) имеют примерно одинаковую относительную погрешность δ, то относительная погрешность выражения r принимается равной $\delta r=\sqrt{3(m+n)}\cdot\delta$, $m+n>10$.

Если у одного из чисел a_i, b_j относительная погрешность значительно превышает относительные погрешности остальных чисел, то относительная погрешность выражения r считается равной этой наибольшей погрешности. При этом в результате целесообразно сохранять столько знаков (значащих цифр), сколько их в числе с наибольшей относительной погрешностью.

Рассмотрим два числа: $a=x+\Delta x$, $b=y+\Delta y$.

Перемножим, левые и правые части соотношений, получим:

$$a\cdot b=(x+\Delta x)(y+\Delta y)=xy+x\cdot\Delta y+\Delta x\cdot y+\Delta x\cdot\Delta y .$$

Переходя к абсолютным величинам правых и левых частей, находим

$$|a\cdot b-x\cdot y|=|x\cdot\Delta y+\Delta x\cdot y+\Delta x\cdot\Delta y| ,$$

по свойству модулей (модуль суммы меньше либо равен сумме модулей) получаем

$$|a\cdot b-x\cdot y|=|x\cdot\Delta y|+|\Delta x\cdot y|+|\Delta x\cdot\Delta y| .$$

Разделим левую и правую части неравенства на $|xy|$, тогда получаем

$$\left|\frac{ab-xy}{xy}\right|\leqslant\left|\frac{\Delta x}{x}\right|+\left|\frac{\Delta y}{y}\right|+\left|\frac{\Delta x\cdot\Delta y}{xy}\right|$$

$\left|\frac{\Delta x}{x}\right|$ – относительная погрешность числа *a*,

$\left|\frac{\Delta y}{y}\right|$ – относительная погрешность числа *b*,

$\left|\frac{\Delta x\cdot\Delta y}{xy}\right|$ – относительная погрешность произведения. В силу его малости отбрасываем. Тогда получаем $\delta ab\leqslant\delta a+\delta b$.

Таким образом, в качестве относительной погрешности произведения можно принять сумму относительных погрешностей сомножителей.

Замечание 1. При умножении приближенного числа на точный сомножитель k относительная погрешность произведения равна относительной погрешности приближенного числа, а абсолютная погрешность в $|k|$ раз больше абсолютной погрешности приближенного числа.

Замечание 2. При перемножении чисел с разной относительной погрешностью (имеющих разное число верных значащих цифр) выполняют следующие действия:

1) выделяют число с наименьшим количеством верных значащих цифр (наименее точное число);
2) округляют оставшиеся сомножители таким образом, чтобы они содержали на одну значащую цифру больше, чем количество верных значащих цифр в выделенном числе;
3) сохраняют в произведении столько значащих цифр, сколько верных значащих цифр имеет наименее точный из сомножителей (выделенное число).

Пример 12. Найти произведение приближенных чисел x_1=12,4 и x_2=65,54 и число верных знаков, если все написанные цифры сомножителей верны в узком смысле.

Решение. Данные числа имеют разное количество цифр после запятой, оставляем эти цифры без изменения. Находим произведение этих чисел: a=12,4·65,54=812,696.

Сохранить нужно три значащих цифры (по правилу), следовательно, получаем, a=813.

Вычислим погрешность:

$$\delta a=\delta x_1+\delta x_2=\frac{0{,}05}{12{,}4}+\frac{0{,}005}{65{,}54}=0{,}0040322+0{,}0000762=$$
$$=0{,}0041084\approx 0{,}0041,$$

$$\Delta a=|a|\delta a=813\cdot 0{,}0041\approx 3\ .$$

Ответ: a=813±3.

Пусть имеем два числа $a=x+\Delta x$ и $b=y+\Delta y$. Вычислим абсолютную погрешность частного:

$$\left|\frac{a}{b}-\frac{x}{y}\right|=\left|\frac{x+\Delta x}{y+\Delta y}-\frac{x}{y}\right|=\left|\frac{y(x+\Delta x)-x(y+\Delta y)}{y(y+\Delta y)}\right|=$$
$$=\left|\frac{yx+y\Delta x-xy-x\Delta y}{y(y+\Delta y)}\right|=\left|\frac{y\Delta x-x\Delta y}{y(y+\Delta y)}\right|.$$

Разделим обе части равенства на $\left|\frac{x}{y}\right|$, получаем

$$\left|\frac{\frac{a}{b}-\frac{x}{y}}{\frac{x}{y}}\right|=\left|\frac{y\Delta x-x\Delta y}{y(y+\Delta y)}\right|\cdot\left|\frac{y}{x}\right|=$$

$$=\left|\frac{y\Delta x-x\Delta y}{x(y+\Delta y)}\right|=\left|\frac{\Delta x}{x}\cdot\frac{y}{y+\Delta y}-\frac{\Delta y}{y}\cdot\frac{y}{y+\Delta y}\right|=$$

$$=\left|\frac{y}{y+\Delta y}\right|\cdot\left|\frac{\Delta x}{x}-\frac{\Delta y}{y}\right|\leqslant\left|\frac{y}{y+\Delta y}\right|\cdot\left(\left|\frac{\Delta x}{x}\right|+\left|\frac{\Delta y}{y}\right|\right).$$

Так как Δy малая величина по сравнению с y, то выражение $\frac{y}{y+\Delta y}\approx 1$, тогда $\delta a/b\leqslant\delta a+\delta b$.

Следовательно, относительная погрешность частного не превышает суммы относительных погрешностей делимого и делителя.

Замечание. При делении чисел с различным числом верных значащих цифр выполняют те же действия, что и при умножении.

Пример 13. Вычислить выражение $r = \dfrac{3{,}2 \cdot 356{,}7 \cdot 0{,}04811}{7{,}1948 \cdot 34{,}56}$,

считая, что все числа даны с верными знаками, т. е. что их абсолютные погрешности не превосходят половины единицы младшего оставленного разряда.

Решение. Наибольшую относительную погрешность имеет число $a = 3{,}2$, которое содержит всего два верных знака (против четырех-пяти верных знаков в остальных числах):

$$\delta a = \frac{0{,}05}{3{,}2} = 0{,}016 = 1{,}6\% .$$

Поэтому можно считать, что относительная погрешность результата составляет $\delta r = 1{,}6\%$, т. е. что результат содержит не более двух верных знаков. Так как количество данных чисел невелико, то в расчетах сохраняем один запасной знак, округляя все числа до трех знаков:

$$r = \frac{3{,}2 \cdot 356{,}7 \cdot 0{,}04811}{7{,}1948 \cdot 34{,}56} = 0{,}221 .$$

Абсолютную погрешность результата вычисляем по его относительной погрешности и найденному численному значению:

$$\Delta r = r \cdot \delta r = 0{,}221 \cdot 0{,}016 = 0{,}0035 .$$

Округляя результат до верных знаков, отбрасываем запасной знак и получаем $r = 0{,}22$.

1.4 Упражнения

1. Округляя следующие числа до трех значащих цифр, определить абсолютную Δ и относительную δ погрешности полученных приближенных чисел.

а) 2,1514, *б*) 0,16152, *в*) 0,01204, *г*) 1,225, *д*) –0,0015281, *е*) –392,85.

2. Определить абсолютную погрешность следующих приближенных чисел по их относительным погрешностям.

а) a=13267, δ=0,1%, *б*) a=2,32, δ=0,7%, *в*) a=35,72, δ=1%, *г*) a=0,896, δ=10%.

3. Определить количество верных знаков в числе x, если известна его абсолютная погрешность.

а) x=0,3941, $\Delta x = 0{,}25\cdot 10^{-2}$, *б*) x=0,1132, $\Delta x = 0{,}1\cdot 10^{-3}$,

в) x=38,2543, $\Delta x = 0{,}27\cdot 10^{-2}$, *г*) x = 293,481, $\Delta x = 0{,}1$.

4. Определить количество верных знаков в числе a, если известна его относительная погрешность.

а) a=1,8921, $\delta = 0{,}1\cdot 10^{-2}$, *б*) a=0,2218, $\delta = 0{,}20\cdot 10^{-1}$,

в) a=22,351, δ=0,1, *г*) a=0,02425, $\delta = 0{,}5\cdot 10^{-2}$,

д) a=0,000135, δ=0,15, *е*) a=9,3598, δ=0,1%.

5. Найти суммы приближенных чисел и указать их погрешности.

а) 0,145 + 321+78,2 (все знаки верные),

б) $x_1+x_2+x_3$, где x_1=197,6, $\Delta x_1 = 0{,}2$, x_2=23,44, $\Delta x_2 = 0{,}22$, x_3=1,55, $\Delta x_3 = 0{,}17$.

6. Вычислить выражения и указать их погрешности, считая в исходных данных все знаки верными.

а) $y=\frac{3{,}07\cdot 326}{36{,}4\cdot 323}$, *б*) $y=\frac{96{,}891-4{,}25}{33{,}3+0{,}426}$.

1.5 Лабораторная работа «Питон для математиков»

Python (в русском языке встречаются названия питóн или пáйтон) – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Освоение языка Python начинается со стандартной библиотеки, этот путь вами уже пройден. На занятиях мы будем использовать библиотеки Math, NumPy, SciPy, SymPy и Mathplotlib для построения графиков.

Библиотека math – предоставляет обширный функционал для работы с числами, содержит основные математические функции.

Таблица 1. Функции в библиотеке Math

функция	**описание**
math.ceil(x)	округление до ближайшего большего числа
math.floor(X)	округление вниз
math.trunc(X)	усекает значение X до целого
math.fabs(X)	модуль X
math.fmod(X, Y)	остаток от деления X на Y
math.factorial(X)	факториал X
math.modf(X)	возвращает целую и дробную часть X, оба числа имеют тот же знак, что и X
math.frexp(X)	возвращает мантиссу и экспоненту
math.exp(X)	e^x
math.log(X, [base])	логарифм X по основанию base. Если base не указан, вычисляется натуральный логарифм
math.log1p(X)	натуральный логарифм (1 + X). При $X \to 0$ точнее, чем math.log(1+X)
math.log2(X)	логарифм X по основанию 2
math.pow(X, Y)	x^y
math.sqrt(X)	квадратный корень из X
math.cos(X)	косинус X (X указывается в радианах)

Окончание табл. 1

функция	**описание**
math.sin(X)	синус X (X указывается в радианах)
math.tan(X)	тангенс X (X указывается в радианах)
math.acos(X)	арккосинус X (в радианах)
math.asin(X)	арксинус X (в радианах)
math.atan(X)	тангенс X (X указывается в радианах)
math.pi	π
math.e	e^x

Библиотека NumPy – Numerical Python. Здесь реализовано множество вычислительных механизмов, пакет поддерживает специализированные структуры данных, в том числе – одномерные и многомерные массивы, значительно расширяющие возможности Python по выполнению различных вычислений. NumPy можно рассматривать как свободную альтернативу MATLAB. Язык программирования MATLAB внешне напоминает NumPy: оба они интерпретируемые, оба позволяют выполнять операции над массивами (матрицами), а не над скалярами. И MATLAB, и NumPy для решения основных задач линейной алгебры используют код, основанный на коде библиотеки LAPACK.

Таблица 2. Команды в библиотеке NumPy

команда	**описание**
import numpy as np	загрузить библиотеку numpy
a = np.array([1, 4, 5, 8], float)	создать одномерный массив из списка с вещественными

Продолжение табл. 2

команда	описание
	значениями
a = np.array([[1, 2, 3], [4, 5, 6]], float)	создание двумерного массива
a.shape	возвращает количество строк и столбцов в матрице
a.dtype	возвращает тип переменных, хранящихся в матрице
a[0], a[0,0]	доступ к элементам массива (матрицы). Помните, что в Питоне нумерация начинается с нуля.
:	позволяет работать со срезами массивов
a[1,:]	второй (!) столбец
a[:,2]	третья (!) строка
a[-1:, -2:]	отрицательными индексами тоже можно пользоваться
np.arange(5, dtype=float) array([0., 1., 2., 3., 4.]) np.arange(1, 6, 2, dtype=int) array([1, 3, 5])	команда arange похожа на команду range, только создает вектор из вещественных или целых чисел по порядку с заданным шагом
np.ones((2,3), dtype=float)	создает матрицу указанной размерности из единиц (тип можно менять)
np.zeros(7, dtype=int)	создает матрицу размера 1x7 из

Продолжение табл. 2

команда	описание
	нулей (тип можно менять)
np.identity(4, dtype=float)	создает единичную матрицу (можно указать тип элементов)
np.eye(4, k=1, dtype=float)	создает матрицу с единицами на k-той диагонали
a.t a.transpose()	транспонирование матрицы
A = np.random.rand(4, 5)	создание матрицы из случайных чисел вещественных чисел из интервала (0,1), размер 4x5
np.random.randint(0, 3, (2, 10))	создание матрицы из случайных целых чисел из полуотрезка [0,3) размер матрицы (2,10)
np.random.uniform(0, 3, (2,10))	создание матрицы из случайных вещественных чисел из полуотрезка [0,3) размер матрицы (2,10)
+, -, *, \	поэлементное выполнение операций над массивами (матрицами)
a = np.array([1, 2, 3], float) b = np.array([0, 1, 1], float) np.dot(a, b)	скалярное умножение строк
np.dot(a, b)	функция dot позволят умножать матрицы
np.linalg.det(a)	вычисляет определитель матрицы a

Окончание табл. 2

команда	описание
np.linalg.eig(a)	собственные векторы и собственные значения
np.set_printoptions (formatter={'float': "{0:.0f}".format})	команда позволяет оформлять вывод с плавающей точкой

Библиотека SciPy очень хорошо расширяет функционал NumPy. В настоящее время библиотека SciPy поддерживает интеграцию, градиентную оптимизацию, специальные функции, средства решения обыкновенных дифференциальных уравнений, инструменты параллельного программирования и многое другое. Другими словами, мы можем сказать, что если что-то есть в общем учебнике числовых вычислений, высока вероятность того, что вы найдете его реализацию в SciPy. В SciPy есть набор пакетов для разных научных вычислений.

Таблица 3. Пакеты в библиотеке SciPy

название пакета	описание пакета
cluster	алгоритмы кластерного анализа
constants	физические и математические константы
fftpack	быстрое преобразование Фурье
integrate	решения интегральных и обычных дифференциальных уравнений
interpolate	интерполяция и сглаживание сплайнов
linalg	линейная алгебра

Окончание табл. 3

название пакета	описание пакета
ndimage	N-размерная обработка изображений
odr	метод ортогональных расстояний
optimize	оптимизация и численное решение уравнений
signal	обработка сигналов
sparse	разреженные матрицы
spatial	разреженные структуры данных и алгоритмы
special	специальные функции
stats	статистические распределения и функции

Задание к лабораторной работе.

Пример выполнения приведен по ссылке:

https://drive.google.com/file/d/1mz3GOdB54WyvrlN7tUVDo9epR0oTYKz4/view?usp=sharing

Вариант 1.

1. Создать матрицу 5x5 случайных целых принадлежащих полуотрезку [0, 10). Транспонировать. Вычислить ее определитель.
2. Создать вектор-столбец и матрицу подходящих размеров. Выполнить умножение матриц.
3. Решить систему уравнений $\begin{cases} x_1 - x_3 = 1 \\ -x_1 - x_2 + 3x_3 = -3 \\ x_1 - 2x_2 - 4x_3 = 5 \end{cases}$.
4. Вычислить интеграл $\int_0^1 \sqrt{x} + \sqrt[3]{x^2}\, dx$.

5. Вычислить интеграл $\int_{-1}^{1} dy \int_{2y}^{y} (x-y)e^{y}\,dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = 3\sin x \\ y = \sqrt{x+5} \end{cases}$.Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 2.

1. Создать матрицу 5x5 случайных вещественных чисел, принадлежащих интервалу (0, 2). Транспонировать. Вычислить ее определитель.

2. Создать вектор-столбец и матрицу подходящих размеров. Выполнить умножение матриц.

3. Найти собственные векторы и собственные значения $A = \begin{pmatrix} 0 & -3 & -1 \\ 3 & 8 & 2 \\ -7 & -15 & -3 \end{pmatrix}$.

4. Вычислить интеграл $\int_{0}^{4} \frac{dx}{1+\sqrt{2x+1}}$.

5. Вычислить интеграл $\int_{0}^{\pi/2} dx \int_{0}^{x} \cos(x+y)\,dx$

6. Построить в одной системе координат графики функций: $\begin{cases} y = \ln(x+5) \\ y = 3x-2 \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 3.

1. Создать матрицу 5x5 из единиц. Создать единичную матрицу 50x50.

2. Вычислить определитель $\begin{vmatrix} 3 & -1 & 2 & 3 & 2 \\ 1 & 2 & -3 & 3 & 4 \\ 2 & -3 & 4 & 2 & 1 \\ 3 & 0 & 0 & 5 & 0 \\ 2 & 0 & 0 & 4 & 0 \end{vmatrix}$.

3. Создать случайную матрицу A из целых чисел из отрезка [0,5] размера 4x4. Создать вектор-столбец B подходящего размера. Решить систему AX = B.

4. Вычислить интеграл $\int_0^{1/3} \mathrm{ch}^2 3x\,dx$.

5. Вычислить интеграл $\int_0^1 dx \int_0^{1-x} dy \int_0^{1-x-y} (x+y+z)\,dz$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = 2\cos(x - \pi/4) \\ y = x + 3 \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 4.

1. Создать матрицу 7x7 случайных целых принадлежащих отрезку [0, 10]. Транспонировать. Вычислить ее определитель.

2. Создать две матрицы подходящих размеров. Выполнить умножение матриц.

3. Решить систему уравнений $\begin{cases} 3x_1 + 2x_2 + x_3 = 5 \\ 3x_1 + 3x_2 + 2x_3 = 7 \\ 5x_1 + 5x_2 + 3x_3 = 11 \end{cases}$.

4. Вычислить интеграл $\int_0^{\pi/4} \frac{dx}{1 + 2\sin^2 x}$.

5. Вычислить интеграл $\int_{-1}^{2} dy \int_{y^2}^{y+2} y^2\,dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = 1 - \cos(x) \\ y = \sqrt{3 - x} \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 5.

1. Создать матрицу 5x5 случайных вещественных чисел, принадлежащих интервалу (-3, 3). Транспонировать. Вычислить ее определитель.

2. Создать вектор-столбец и матрицу подходящих размеров. Выполнить умножение матриц.

3. Найти собственные векторы и собственные значения $A = \begin{pmatrix} -7 & -5 & -5 \\ 0 & 3 & 0 \\ 10 & 5 & 8 \end{pmatrix}$.

4. Вычислить интеграл $\int_0^{\pi/2} e^{2x} cosx\, dx$.

5. Вычислить интеграл $\int_{-\infty}^{+\infty} \frac{dx}{x^2 + 4x + 9}$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = \ln(x) + 2 \\ y = -3x \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 6.

1. Создать матрицу 10x10 из вещественных единиц. Создать единичную матрицу 10x10.

2. Вычислить определитель $\begin{vmatrix} 2 & 1 & 3 & 6 \\ 4 & 1 & 3 & 3 \\ 5 & 2 & 4 & 1 \\ 5 & 1 & 2 & 2 \end{vmatrix}$.

3. Создать случайную матрицу A из целых чисел из отрезка [-3,5] размера 4x4. Создать вектор-столбец B подходящего размера. Решить систему AX = B.

4. Вычислить интеграл $\int\limits_{-1/2}^{1/2} \frac{dx}{\sqrt{1-x^2}}$.

5. Вычислить интеграл $\int\limits_{0}^{+\infty} \cos 2x \, dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = \sin(x + \pi/3) \\ y = 2x \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 7.

1. Создать матрицу 5x5 случайных целых принадлежащих полуотрезку [0, 10). Транспонировать. Вычислить ее определитель.

2. Создать вектор-столбец и матрицу подходящих размеров. Выполнить умножение матриц.

3. Решить систему уравнений $\begin{cases} x_1 + 2x_2 - x_3 = 2 \\ 3x_1 - x_2 + x_3 = 3 \\ 2x_1 + x_2 - 4x_3 = -1 \end{cases}$.

4. Вычислить интеграл $\int_0^1 \sqrt{x} + \sqrt[3]{x^2} \, dx$.

5. Вычислить интеграл $\int_{-1}^{1} dy \int_{2y}^{y} (x - y) e^y \, dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = -2\cos x \\ y = \sqrt{x+1} \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 8.

1. Создать матрицу 5x5 случайных вещественных чисел, принадлежащих интервалу (0, 2). Транспонировать. Вычислить ее определитель.

2. Создать вектор-столбец и матрицу подходящих размеров. Выполнить умножение матриц.

3. Найти собственные векторы и собственные значения $A = \begin{pmatrix} 2 & 1 & -3 \\ 0 & 1 & -1 \\ 0 & -2 & 2 \end{pmatrix}$.

4. Вычислить интеграл $\int_0^4 \frac{dx}{1+\sqrt{2x+1}}$.

5. Вычислить интеграл $\int_0^{\pi/2} dx \int_0^x \cos(x+y)\,dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = \ln(2-x) \\ y = -x/2 \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 9.

1. Создать матрицу 5x5 из единиц. Создать единичную матрицу 50x50.

2. Вычислить определитель $\begin{vmatrix} -1 & 3 & 4 & 0 \\ 2 & 3 & 4 & -2 \\ 3 & 2 & 1 & -5 \\ 4 & 3 & 1 & 2 \end{vmatrix}$.

3. Создать случайную матрицу A из целых чисел из отрезка [0,5] размера 4x4. Создать вектор-столбец B подходящего размера. Решить систему AX = B.

4. Вычислить интеграл $\int_0^{1/3} \mathrm{ch}^2 3x\,dx$.

5. Вычислить интеграл $\int_0^1 dx \int_0^{1-x} dy \int_0^{1-x-y} (x+y+z)\,dz$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = 2\sin(x+\pi/3) \\ y = x-1 \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

Вариант 10.

1. Создать матрицу 7x7 случайных целых принадлежащих отрезку [0, 10]. Транспонировать. Вычислить ее определитель.

2. Создать две матрицы подходящих размеров. Выполнить умножение матриц.

3. Решить систему уравнений $\begin{cases} 3x_1 - 2x_2 + x_3 = 1 \\ -5x_1 + 3x_2 - 2x_3 = -3 \\ x_1 + x_2 + 2x_3 = 7 \end{cases}$.

4. Вычислить интеграл $\int_0^{\pi/4} \frac{dx}{1+2\sin^2 x}$.

5. Вычислить интеграл $\int_{-1}^{2} dy \int_{y^2}^{y+2} y^2\,dx$.

6. Построить в одной системе координат графики функций: $\begin{cases} y = 1 + \sin(x) \\ y = \sqrt{2 + x} \end{cases}$. Оси координат должны быть подписаны, графики должны быть разного цвета, должна быть выведена легенда. Точку пересечения (если она есть) отметить на графике.

2 ВЕКТОРЫ И МАТРИЦЫ

2.1 Норма вектора

Решением системы линейных алгебраических уравнений является вектор $x=(x_1, x_2, \dots, x_m)^T$, который будем рассматривать как элемент векторного пространства $\mathbb{R}^m$. Приближенное решение $x^*=(x_1^*, x_2^*, \dots, x_m^*)^T$ и погрешность $e=x-x*=(x_1-x_1^*, \dots, x_m-x_m^*)^T$ также являются элементами $\mathbb{R}^m$. Для того чтобы анализировать методы решения систем, необходимо уметь количественно оценивать «величины» векторов $x*$ и $x-x*$, а также векторов $b*$ и $b-b*$, где $b^*=(b_1^*, b_2^*, \dots, b_m^*)^T$ вектор приближенно заданных правых частей. Удобной для этой цели количественной характеристикой является широко используемое понятие нормы вектора.

Говорят, что в $\mathbb{R}^m$ задана норма, если каждому вектору x из $\mathbb{R}^m$ сопоставлено вещественное число $\|x\|$, называемое нормой вектора x и обладающее следующими свойствами:

1. $\|x\|\geqslant 0$, причем $\|x\|=0$ тогда и только тогда, когда $x=0$;
2. $\|\alpha x\|=|\alpha|\cdot\|x\|$ для любого вектора x и любого числа α;
3. $\|x+y\|\leqslant\|x\|+\|y\|$ для любых векторов x и y.

Заметим, что такими же свойствами обладает обычная геометрическая длина вектора в трехмерном пространстве. Свойство 3 в этом случае следует из правила сложения векторов и из того известного факта, что сумма длин двух сторон треугольника всегда больше длины третьей стороны.

Существует множество различных способов введения норм. В вычислительных методах наиболее употребительными являются следующие три нормы:

$$\|x\|_1 = \sum_{i=1}^{m} |x_i| , \quad \|x\|_2 = |x| = \left(\sum_{i=1}^{m} |x_i|^2 \right)^{\frac{1}{2}} , \quad \|x\|_\infty = \max_{1 \leqslant i \leqslant m} |x_i| .$$

Первые две из них являются частными случаями более общей нормы:

$$\|x\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{\frac{1}{p}} , \quad p \geq 1$$

при $p=1$, $p=2$, а последняя получается предельным пе-реходом при $p \to +\infty$.

Норма $\|x\|_2 = |x|$ является естественным обобщением на случай m-мерного пространства понятия длины вектора в двух- и трехмерных геометрических пространствах. Поэтому ее называют евклидовой нормой или модулем вектора x.

Справедливы неравенства $\|x\|_\infty \leqslant \|x\|_2 \leqslant \|x\|_1 \leqslant m\|x\|_\infty$, указывающие на то, что в определенном смысле все три введенные нормы эквивалентны: каждая из них оценивается любой из двух других норм с точностью до множителя, зависящего от m.

Пример 1. Для вектора $(0{,}12\,,\ -0{,}15\,,\ \ 0{,}16)^T$ вычислить $\|x\|_1$, $\|x\|_2$, $\|x\|_\infty$.

Решение. $\|x\|_1 = \sum_{i=1}^{m} |x_i| = 0{,}12 + 0{,}15 + 0{,}16 = 0{,}43$,

$$\|x\|_2 = \left(\sum_{i=1}^{m} |x_i|^2 \right)^{\frac{1}{2}} = (0{,}12^2 + 0{,}15^2 + 0{,}16^2)^{\frac{1}{2}} = 0{,}25 ,$$

$$\|x\|_\infty = \max_{1 \leqslant i \leqslant m} |x_i| = \max\{0{,}12\,,\ \ 0{,}15\,,\ \ 0{,}16\} = 0{,}16 .$$

Скалярным произведением векторов $x=(x_1, x_2, \dots, x_m)^T$ и $y=(y_1, y_2, \dots, y_m)^T$ называется величина

$$(x, y)=x_1 y_1+x_2 y_2+\dots+x_m y_m=\sum_{i=1}^{m} x_i y_i .$$

Нетрудно заметить, что $\|x\|_2=(x, x)^{\frac{1}{2}}$.

Когда векторы x, y имеют комплексные компоненты, скалярное произведение понимают так:

$$(x, \bar{y})=x_1 \bar{y}_1+x_2 \bar{y}_2+\dots+x_m \bar{y}_m=\sum_{i=1}^{m} x_i \bar{y}_i .$$

Будем всюду считать, что в пространстве m-мерных векторов $\mathbb{R}^m$ введена и фиксирована некоторая норма $\|x\|$. В этом случае в качестве меры степени близости векторов x и x^* естественно использовать величину $\|x-x^*\|$, являющуюся аналогом расстояния между точками x и x^*. Введем абсолютную и относительную погрешности вектора x^* с помощью формул $\Delta(x^*)=\|x-x^*\|$, $\delta(x^*)=\dfrac{\|x-x^*\|}{\|x\|}$.

Выбор той или иной конкретной нормы в практических задачах диктуется тем, какие требования предъявляются к точности решения.

Выбор нормы $\|x\|_1$ фактически отвечает случаю, когда малой должна быть суммарная абсолютная погрешность в компонентах решения; выбор $\|x\|_2$ соответствует критерию малости среднеквадратичной погрешности, а принятие в качестве нормы $\|x\|_\infty$ означает, что малой должна быть максимальная из абсолютных погрешностей в компонентах решения.

Пусть $\{x^{(n)}\}_{n=1}^{\infty}$ – последовательность векторов $x^{(n)}=(x_1^{(n)}, x_2^{(n)}, \dots, x_m^{(n)})^T$. Говорят, что последовательность векторов $x^{(n)}$ сходится к вектору x ($x^{(n)} \to x$ при $n \to \infty$), если $\Delta(x^{(n)})=\|x^{(n)}-x\| \to 0$ при $n \to \infty$.

Сам факт наличия или отсутствия сходимости $x^{(n)}$ к x при $n \to \infty$ в конечномерных пространствах не зависит от выбора нормы. Известно, что из сходимости последовательности по одной из норм следует сходимость этой последовательности по любой другой норме. Более того, $x^{(n)} \to x$ при $n \to \infty$ тогда и только тогда, когда для всех $i=1,2,\dots,m$ имеем $x_i^{(n)} \to x_i$, при $n \to \infty$, т. е. сходимость по норме в $\mathbb{R}^m$ эквивалентна покомпонентной (по координатной) сходимости.

2.2 Норма матрицы

Величина $\|A\|=\max\limits_{x \neq 0} \dfrac{\|Ax\|}{\|x\|}$ называется нормой матрицы A, подчиненной норме векторов, введенной в $\mathbb{R}^m$.

Заметим, что множество всех квадратных матриц размера $m \times m$ является векторным пространством. Можно показать, что введенная в этом пространстве норма обладает следующими свойствами, аналогичными свойствам нормы вектора:

1) $\|A\| \geqslant 0$, причем $\|A\|=0$ тогда и только тогда, когда A=0;

2) $\|\alpha A\|=|\alpha| \cdot \|A\|$ для любой матрицы A и любого числа α;

3) $\|A+B\| \leqslant \|A\|+\|B\|$ для любых матриц A и B;

4) $\|A \cdot B\| \leqslant \|A\| \cdot \|B\|$ для любых матриц A и B;

5) для любой матрицы A и любого вектора x справедливо неравенство $\|Ax\| \leqslant \|A\| \cdot \|x\|$.

Докажем, например, свойство 5. Если $\|x\| \neq 0$, то неравенство эквивалентно неравенству $\dfrac{\|Ax\|}{\|x\|} \leqslant \|Ax\|$, справедливость которого следует из определения нормы. Если же $\|x\| = 0$, то неравенство превращается в верное числовое неравенство $0 \leqslant 0$.

Как следует из определения, каждой из векторных норм $\|x\|$ соответствует своя подчиненная норма матрицы A. Известно, в частности, что нормам $\|x\|_1$, $\|x\|_2$ и $\|x\|_\infty$ подчинены нормы $\|A\|_1$, $\|A\|_2$ и $\|A\|_\infty$, вычисляемые по формулам

$$\|A\|_1 = \max_{1 \leqslant j \leqslant m} \sum_{i=1}^{m} |a_{ij}|, \quad \|A\|_2 = \max_{1 \leqslant j \leqslant m} \sqrt{\lambda_j(A^T A)},$$

где $\lambda_j(A^T A)$ собственные числа матрицы $\lambda_j(A^T A)$. Число λ_j называется собственным числом матрицы A, если существует вектор $x \neq 0$ такой, что $Ax = \lambda x$. Каждая матрица порядка m имеет ровно m собственных чисел (вообще говоря, комплексных) с учетом их кратности.

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{m} |a_{ij}|.$$

Нормы $\|A\|_1$ и $\|A\|_\infty$, вычисляются просто. Для получения значения первой из них нужно найти сумму модулей элементов каждого из столбцов матрицы A, а затем выбрать максимальную из этих сумм. Для получения значения $\|A\|_\infty$ нужно аналогичным образом поступить со строками матрицы A. Как правило, вычислить значение нормы $\|A\|_2$ бывает трудно, так как для этого следует искать собственные числа λ_j. Для оценки величины

$\|A\|_2$ можно, например, использовать неравенство $\|A\|_2 \leqslant \|A\|_E$.

$\|A\|_E = \sqrt{\sum_{i,j=1}^{m} |a_{ij}|^2}$ – величина, называемая евклидовой нормой матрицы A. Часто эту же величину называют нормой Фробениуса и обозначают $\|A\|_F$.

Норма $\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$ имеет простую геометрическую интерпретацию. Для того чтобы ее привести, заметим, что операцию умножения матрицы A на вектор x можно рассматривать как преобразование, которое переводит вектор x в новый вектор $y = Ax$. Если значение $\|x\|$ интерпретируется как длина вектора x, то величина $\frac{\|Ax\|}{\|x\|}$ есть коэффициент растяжения вектора x под действием матрицы A. Таким образом, величина

$$k_{\max} = \|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

представляет собой максимальный коэффициент растяжения векторов под действием матрицы A. Полезно отметить, что для невырожденной матрицы A минимальный коэффициент растяжения $k_{\min}$ отвечает норме обратной матрицы и вычисляется по формуле

$$k_{\min} = \|A^{-1}\|^{-1} = \min_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Заметим, что в случае $\|A\| < 1$ происходит сжатие векторов под действием матрицы A.

Пример 2. Для матрицы $A=\begin{pmatrix} 0{,}1 & -0{,}4 & 0 \\ 0{,}2 & 0 & -0{,}3 \\ 0 & 0{,}1 & 0{,}3 \end{pmatrix}$ найти $\|A\|_1$ и $\|A\|_\infty$ и оценить $\|A\|_2$.

Решение.
$$\|A\|_1 = \max_{1 \leqslant j \leqslant m} \sum_{i=1}^{m} |a_{ij}| =$$
$$= \max\{0{,}1+0{,}2+0;\ 0{,}4+0+0{,}1;\ 0+0{,}3+0{,}3\} = 0{,}6$$

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{m} |a_{ij}| =$$
$$= \max\{0{,}1+0{,}4+0;\ 0{,}2+0+0{,}3;\ 0+0{,}1+0{,}3\} = 0{,}5$$

$$\|A\|_2 \leqslant \|A\|_E = \sqrt{\sum_{i,j=1}^{m} |a_{ij}|^2} = \sqrt{\sum_{i,j=1}^{3} |a_{ij}|^2} =$$
$$= \sqrt{0{,}01+0{,}16+0+0{,}04+0+0{,}09+0+0{,}01+0{,}09} = \sqrt{0{,}4} \approx 0{,}63 .$$

2.3 Типы используемых матриц

Эффективность вычислений в линейной алгебре существенно зависит от умения использовать специальную структуру и свойства используемых в расчетах матриц.

Квадратная матрица A называется диагональной, если ее элементы удовлетворяют условию $a_{ij}=0$ для $i \neq j$ (все отличные от нуля элементы расположены на главной диагонали).

Квадратная матрица A называется нижней треугольной, если все ее элементы, расположенные выше главной диагонали, равны нулю $a_{ij}=0$ для $i<j$. Если же равны нулю все элементы матрицы, расположенные ниже главной диагонали, то она называется верхней треугольной.

Квадратная матрица A называется симметричной, если она совпадает со своей транспонированной матрицей $A^{\text{т}}$ ($a_{ij}=a_{ji}$, для всех i, j).

Симметричную матрицу A называют положительно определенной, если для всех векторов $x\neq 0$ квадратичная форма

$$(Ax\,,x)=\sum_{i\,,\,j=1}^{m} a_{ij}x_i x_j$$

принимает положительные значения.

Обозначим через $\lambda_{\max}$ и $\lambda_{\min}$ максимальное и минимальное собственные значения матрицы A. Известно, что для симметричной матрицы

$$\lambda_{\min}\|x\|_2^2\leqslant(Ax\,,x)\leqslant\lambda_{\max}\|x\|_2^2$$

и матрица A положительно определена тогда и только тогда, когда все ее собственные значения положительны.

Одна из трудностей практического решения систем большой размерности связана с ограниченностью оперативной памяти компьютера. Хотя объем оперативной памяти вновь создаваемых вычислительных машин растет очень быстро, тем не менее еще быстрее возрастают потребности практики в решении задач все большей размерности (для хранения в оперативной памяти компьютера матрицы порядка m требуется m^2 машинных слов). В значительной степени ограничения на размерность решаемых систем можно снять, если использовать для хранения матрицы внешние запоминающие устройства. Однако, в этом случае, многократно возрастают как затраты машинного времени, так и сложность соответствующих алгоритмов. Поэтому, при создании вычислительных алгоритмов линейной алгебры, большое внимание уделяют способам компактного размещения элементов матриц в памяти компьютера.

К счастью, приложения очень часто приводят к матрицам, в которых число ненулевых элементов много меньше общего числа элементов матрицы. Такие матрицы принято называть разреженными. Напротив, матрицы общего вида называют плотными (или заполненными). Многие приложения приводят к системам уравнений с так называемыми ленточными матрицами. Матрица A называется ленточной с полушириной ленты, равной l, если $a_{ij}=0$ для $|i-j|>l$. Все ненулевые элементы такой матрицы расположены на $s=2l+1$ ближайших к главной диагоналях матрицы; число s принято называть шириной ленты. Частным случаем ленточной матрицы при $s=3$ является трехдиагональная матрица. В случае $s \ll m$ ленточная матрица является разреженной.

2.4 Лабораторная работа «Нормы, разложение матриц»

Срезы в массиве.

Матрица в NumPy реализована как двумерный массив ndarray. Ndarray является многомерным однородным массивом с заранее заданным количеством элементов. Однородный — потому что все объекты в нем одного размера или типа. Количество размерностей и объектов массива определяются его размерностью (shape), кортежем n положительных целых чисел. Они указывают размер каждой размерности. Размерности определяются как оси. Размер массивов NumPy фиксирован, а это значит, что после создания объекта его уже нельзя поменять. Это поведение отличается от такового у списков Python, которые могут увеличиваться и уменьшаться в размерах.

Подробнее о массивах смотри

https://pythonru.com/biblioteki/biblioteka-numpy-ndarray-sozdanie-massiva-i-tipy-dannyh

При работе с индексами массивов всегда используются квадратные скобки ([ ]). С помощью индексирования можно ссылаться на отдельные элементы, выделяя их или даже меняя значения. При создании нового массива шкала с индексами создается автоматически.

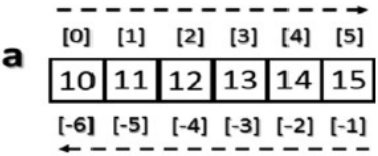

Рисунок 1. Шкала индексов

В зависимости от части массива, которую необходимо извлечь, нужно использовать синтаксис среза; это последовательность числовых значений, разделенная двоеточием (:) в квадратных скобках. Синтаксис: a[start:stop:step].

Чтобы лучше понять синтаксис среза, необходимо рассматривать и случаи, когда явные числовые значения не используются. Если не ввести первое число, NumPy неявно интерпретирует его как 0 (то есть, первый элемент массива). Если пропустить второй он будет заменен на максимальный индекс, а если последний представлен как 1. То есть, все элементы будут перебираться без интервалов.

Таблица 4. Работа с индексами в массиве

команда	**результат**
import numpy as np	загрузили модуль NumPy
a = np.array([10, 11, 12, 13, 14, 15], int)	создание массива из целых чисел out: array([10, 11, 12, 13, 14, 15])
a[2]	получаем второй элемент массива

Продолжение табл. 4

команда	результат
	(помним про нумерацию с нуля) out: 12
a[:]	срез из всех элементов out: array([10, 11, 12, 13, 14, 15])
a[:2]	срез до второго элемента out: array([10, 11])
a[2:]	срез от второго элемента и до конца out: array([12, 13, 14, 15])
a[2:4]	срез от второго до четвертого элемента, правую границу не включаем out: array([12, 13])
a[::2]	срез с шагом 2 out: array([10, 12, 14])
a[::-1]	срез с шагом -1 out: array([15, 14, 13, 12, 11, 10])
a[1:5:2]	срез от первого до пятого элемента с шагом 2 out: array([11, 13])
a[-1]	отрицательные индексы тоже в ходу, получим последний элемент out: 15
a[-3:-1]	out: array([13, 14])
a[5:1:-1]	out: array([15, 14, 13, 12])
b = np.array([[10, 11, 12, 13], [20, 21, 22, 23], [30, 31, 32, 33], [40, 41, 42, 43]], int)	такая конструкция позволяет получать матрицы out: array([[10, 11, 12, 13], [20, 21, 22, 23],

Окончание табл. 4

команда	результат
	[30, 31, 32, 33], [40, 41, 42, 43]])
b[2,1]	получаем элемент в третьей строке втором столбце(помним про нумерацию с нуля) out: 31
b[:, 1]	второй столбец out: array([11, 21, 31, 41])
b[2,:]	out: array([30, 31, 32, 33])
b[1:4:2,:]	out: array([[20, 21, 22, 23], [40, 41, 42, 43]])
b[1:4:2,::2]	out: array([[20, 22],[40, 42]])

Умножение матриц.

Операция умножения матриц вам хорошо известна из курса линейной алгебры. Умножение матрицы A на матрицу B возможно только в случае, когда количество столбцов матрицы A совпадает с количеством строк матрицы B. Элемент c_{ij} новой матрицы получается как умножение i-той строки матрицы A на j-тый столбец матрицы B.

Возможно несколько реализаций матричного умножения: в скалярном виде, в векторном и в матричном.

Скалярный вид: $c_{ij}=\sum_{p=1}^{k} a_{ip} b_{pj}$, где матрица A имеет размер $n\times k$, матрица B имеет размер $k\times m$.

Реализация умножения матриц в скалярном виде представлена на рисунке ниже.

```
n = 3
k = 4
m = 5
A =  np.random.randint(0,11,(n, k))
B = np.random.randint(0,11,(k, m))
C = np.zeros((n,m), dtype = int)
```

```
for i in range(n):
    for j in range(m):
        for p in range(k):
            C[i,j] = C[i,j] + A[i,p]*B[p,j]
```

Рисунок 2. Скалярное (поэлементное) умножение матриц

Однако такое решение является достаточно медленным. Векторные операции существенно увеличивают скорость вычислений. Реализация умножения матриц в векторном виде позволяет сразу находить произведение *i*-той строки матрицы *A* на *j*-тый столбец матрицы *B*.

Векторный вид: $c_{ij} = A[i,:] \cdot B[:,j], 1 \le i \le n, 1 \le j \le m$.

Напомню, что в этом случае операция умножения – это уже скалярное умножение из модуля NumPy (np.dot).

Мы можем использовать также матричное умножение. В нашем случае будем «собирать» матрицу *C* по строкам (для нахождения *i*-той строки матрицы *C* умножим *i*-тую строку матрицы *A* на матрицу *B*).

Матричный вид: C[i, :] = A[i, :]·B .

Аналогично можно «собрать» матрицу *C* по столбцам: C[:, j] = =A·B[:,j].

Приведение матрицы A к *LU* виду.

LU разложение – это представление матрицы *A* в виде произведения двух матриц: *L* – нижняя унитреугольная, *U* – верхняя треугольная. LU разложение используется для решения систем линейных уравнений, обращения матриц и вычисления определителя. *LU* разложение существует только в том случае, когда матрица

A обратима (невырождена), и все ведущие (угловые) главные миноры матрицы A невырождены.

Допустим, нам удалось найти матрицы L и U такие, что $A = LU$. Очевидно, это возможно только в случае невырожденной матрицы A. Тогда $a_{11}=u_{11}\cdot l_{11}$. Предположим, что $a_{11}=0$, но тогда $l_{11}=0$ или $u_{11}=0$, что означает равенство нулю первой строки матрицы L или первого столбца матрицы U, из чего следует их вырожденность. Итак, далее считаем, что $a_{11}\neq 0$. Представим матрицу A как блочную матрицу: $A=\begin{pmatrix} a_{11} & w \\ v & A' \end{pmatrix}$, где w – вектор-строка $w=(a_{12},\dots a_{1n})$ размера $1\times(n-1)$, v – вектор-столбец $v=\begin{pmatrix} a_{21} \\ \vdots \\ a_{n1} \end{pmatrix}$ размера $(n-1)\times 1$, A' – матрица (минор) размера $(n-1)\times(n-1)$. Аналогичным образом представим матрицы L и U:

$$L=\begin{pmatrix} 1 & 0 \\ v_l & L' \end{pmatrix},\ U=\begin{pmatrix} a_{11} & w_U \\ 0 & U' \end{pmatrix}.$$

Тогда

$$A=LU=\begin{pmatrix} 1 & 0 \\ v_l & L' \end{pmatrix}\cdot\begin{pmatrix} a_{11} & w_U \\ 0 & U' \end{pmatrix}=\begin{pmatrix} a_{11} & w_U \\ v_l\cdot a_{11} & v_l\cdot w_U + L'U' \end{pmatrix}=\begin{pmatrix} a_{11} & w \\ v & A' \end{pmatrix}.$$

Заметим, что умножение $v_l\cdot w_U$ это умножение столбца размера $(n-1)\times 1$ на строку размера $1\times(n-1)$, что влечет получение матрицы размера $(n-1)\times(n-1)$.

Итак, получили формулы нахождения матриц L и U: $w_U=w$, $v_l=v/a_{11}$ (вектор-столбец делится на число), $L'\cdot U'=A'-v_l\cdot w_U$.

Итак, нахождение LU разложения для матрицы A сведено к нахождению LU разложения для матрицы $L'\cdot U'$ размера $(n-1)\times(n-1)$.

Выражение $L' \cdot U' = A' - v_l \cdot w_U = A' - v \cdot w / a_{11}$ называется дополнением Шура элемента a_{11} в матрице A.

Приведем реализацию нахождения LU разложения.

```
U = np.zeros((n,n), float)
L = np.identity(n, float)

for i in range(n):
    for j in range(n):
        if i<=j:
            U[i,j] = A1[i,j] -  np.dot(L[i, :i], U[ :i, j])
        if i>j:
            L[i,j] = (A1[i,j] -  np.dot(L[i, :j], U[ :j, j]) )/ U[j,j]
```

Рисунок 3. LU разложение матрицы A

Норма векторов и матриц.

Напомню определение нормы в векторном пространстве X. Нормой называется такое отображение векторного пространства X во множество вещественных чисел $\mathbb{R}$.

$\|.\| : X \to \mathbb{R}$, что выполняются следующие свойства для любых элементов (векторов или матриц) x, y векторного пространства X и скаляра λ:

1) $\|x\| \geq 0, \|x\| = 0 \Leftrightarrow x = 0$ (полож. определенность);

2) $\|\lambda \cdot x\| = |\lambda| \cdot \|x\|$ (однородность);

3) $\|x + y\| \leq \|x\| + \|y\|$ (неравенство треугольника).

Норма является естественным обобщением понятия длины вектора в евклидовом пространстве, таким образом, нормированные пространства – векторные пространства, оснащённые возможностью определения длины вектора.

Существует множество различных способов введения норм. В вычислительных методах наиболее употребительными являются следующие три нормы:

$$\|x\|_1=\sum_{i=1}^{n}|x_i|\ ,\ \ \|x\|_2=\sqrt{\sum_{i=1}^{n}|x|^2}\ ,\ \ \|x\|_\infty=max_i|x_i|\ .$$

Выбор той или иной конкретной нормы в практических задачах диктуется тем, какие требования предъявляются к точности решения. Выбор нормы $\|x\|_1=\sum_{i=1}^{n}|x_i|$ фактически отвечает случаю, когда малой должна быть суммарная абсолютная погрешность в компонентах решения; выбор $\|x\|_2=\sqrt{\sum_{i=1}^{n}|x|^2}$ соответствует критерию малости среднеквадратичной погрешности, а принятие в качестве нормы $\|x\|_\infty=max_i|x_i|$ означает, что малой должна быть максимальная из абсолютных погрешностей в компонентах решения.

Норма матрицы – норма в линейном пространстве матриц. Обычно, от матричной нормы требуют выполнение условия субмультипликативности: $\|A\cdot B\|\le\|A\|\cdot\|B\|$ для всех матриц A и B. Все нормы, которые рассматриваем в этой лабораторной работе, удовлетворяют условию субмультипликативности. Матричную норму, определенную соотношением $\|A\|=max_{\|x\|\neq 0}\dfrac{\|Ax\|}{\|x\|}$ называют нормой, подчиненной векторной норме $\|x\|$. Можно доказать, что $\|A\|_1=max_j\sum_{i=1}^{n}|a_{ij}|$ – максимум из суммы модулей по всем столбцам, $\|A\|_2=max_j\sqrt{\lambda_j(A^TA)}$ максимум из собственных

значений матрицы $A^T \cdot A$, $\|A\|_\infty = max_i \sum_{j=1}^{n} |a_{ij}|$ максимум из суммы модулей по всем столбцам, $\|A\|_F = \sqrt{\sum_{i,\,j=1}^{n} |a_{ij}|^2}$ – норма Фробениуса.

Для нахождения нормы вектора и матрицы в среде программирования Python можно использовать функцию np.linalg.norm(). Синтаксис:

numpy.linalg.norm(x, ord=None, axis=None, keepdims=False).

В зависимости от параметра ord данная функция может возвращать одну из восьми норм или одну из бесконечного числа векторных норм.

Приведу три из них:

- ord = None – для матриц возвращается норма Фробениуса, для векторов соответствует ord = 2 (то есть вычисляет корень квадратный из суммы квадратов векторов) (установлен по умолчанию);
- ord =np.inf – для матриц возвращается np.max(np.sum(np.abs(x), axis=1)), для векторов возвращается np.max(np.abs(x));
- ord = 1 – для матриц возвращается np.max(np.sum(np.abs(x), axis=0)), для векторов возвращается np.sum(np.abs(x))

Более подробно смотри

https://pyprog.pro/linear_algebra_functions/linalg_norm.html

***QR* разложение матрицы методом Грама-Шмидта.**

Везде далее будем считать, что A является невырожденной квадратной матрицей. Матрица A размера $n \times n$ с комплексными элементами может быть представлена в виде: $A=QR$, где Q –

унитарная матрица размера $n \times n$, а R – верхнетреугольная матрица размера $n \times n$.

В случае, когда матрица A состоит из вещественных чисел, её можно представить в виде $A=QR$, где Q является ортогональной матрицей размера $n \times n$, а R – верхнетреугольная матрица размера $n \times n$.

По аналогии, можно определить варианты этого разложения: QL-, RQ-, и LQ-разложения, где L – нижнетреугольная матрица.

Напомню, что матрица A является ортогональной матрицей, тогда и только тогда, когда выполняется одно из следующих эквивалентных условий:

1) строчки матрицы A образуют ортонормированный базис;
2) столбцы матрицы A образуют ортонормированный базис;
3) $A^{-1} = A^T$;
4) $A^{-1} \cdot A^T = E$.

Нахождение QR разложения невырожденной матрицы A является эквивалентом метода ортогонализации Грама-Шмидта. Если рассматривать столбцы (строки) матрицы A как линейно независимую систему векторов, то к этой системе можно применить метод ортогонализации Грама-Шмидта, и получить матрицу, состоящую из векторов ортонормированного базиса (матрицу Q). Тогда для нахождения матрицы R достаточно найти произведение $Q^{\mathrm{T}}A$ (это следует из того, что если $A=QR$, то $Q^{-1} \cdot A = R \Rightarrow R = Q^T \cdot A$).

Приведу одну из возможных реализаций этого метода.

```
A = np.array([[6, 5, 0],[5, -1, 4],[5, 1, -14],[0, 4, 3]],dtype=float)
Q=np.zeros_like(A)
cnt = 0
for a in A.T:
    u = np.copy(a)
    for i in range(0, cnt):
        u -= np.dot(np.dot(Q[:, i].T, a), Q[:, i]) #метод Грама-Шмидта
    e = u / np.linalg.norm(u)  # Нормализован
    Q[:, cnt] = e
    cnt += 1
R = np.dot(Q.T, A)
```

```
np.set_printoptions(precision=4, suppress=True) #оставляем 4 знака после запятой
print(Q, R)
```

```
[[ 0.647   0.5096  0.1799]
 [ 0.5392 -0.4811  0.5743]
 [ 0.5392 -0.1305 -0.7902]
 [ 0.      0.7013  0.1162]] [[ 9.2736  3.235  -5.3916]
 [ 0.      5.7039  2.006 ]
 [-0.      0.     13.7079]]
```

Рисунок 4. Реализация метода Грама-Шмидта

Метод отражений (метод Хаусхолдера).

Метод Хаусхолдера один из самых распространенных методов нахождения *QR* разложения матрицы *A*. Пусть *v* – *n*-мерный ненулевой вектор-столбец единичной длины (т.е. $\|v\|=1$), *H* – матрица $n\times n$ $H=E-2v\cdot v^T$ называется отражением Хаусхолдера или просто отражением. Можно проверить, что матрица *H* симметрична и ортогональна. Умножению матрицы *H* на произвольный вектор *x* можно дать следующую геометрическую интерпретацию: вектор *Hx* получается отражением вектора *x* относительно гиперплоскости, ортогональной вектору *v*.

Приведем пример. Рассмотрим вектор $x=(1,3,4)^T$, в котором требуется обнулить x_3 . Поскольку x_2 первый элемент отличный от нуля, то матрица Хаусхолдера принимает вид: $H=\begin{pmatrix}1 & 0\\ 0 & H'_{2x2}\end{pmatrix}$. Находим коэффициент: $\beta=sign(-x_2)\|y\|=-5$, где $sign(-x_2)$ знак, противоположный знаку второй координаты вектора x, $\|y\|$ – вторая норма «усеченного» вектора x $y=(3,4)$, $\|y\|=\sqrt{9+16}$.

Тогда $u=(y_1-\beta, y_2)=(8,4)$ и $v=\frac{u}{\|u\|}=\frac{(8,4)^T}{\sqrt{64+16}}=\frac{(2,1)^T}{\sqrt{5}}$.

Находим матрицу

$$H'=E-2vv^T=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}-\begin{pmatrix}2/\sqrt{5}\\ 1/\sqrt{5}\end{pmatrix}\cdot\begin{pmatrix}2/\sqrt{5} & 1/\sqrt{5}\end{pmatrix}=\begin{pmatrix}-0.6 & -0.8\\ -0.8 & 0.6\end{pmatrix}.$$

Результат: $x'=Hx=\begin{pmatrix}1 & 0 & 0\\ 0 & -0.6 & -0.8\\ 0 & -0.8 & 0.6\end{pmatrix}\cdot\begin{pmatrix}1\\ 3\\ 4\end{pmatrix}=\begin{pmatrix}1\\ -5\\ 0\end{pmatrix}$.

Проверка: $\|x'\|=\|x\|=\sqrt{26}$ (везде имеем в виду вторую норму).

Приведу реализацию метода Хаусхолдера на языке Python.

```
A = np.array([[6, 5, 0],[5, -1, 4],[5, 1, -14],[0, 4, 3]],dtype=float)
(r, c) = np.shape(A)
Q = np.identity(r)
R = np.copy(A)
for cnt in range(r - 1):
    x = R[cnt:, cnt]
    e = np.zeros_like(x)
    e[0] = np.linalg.norm(x)
    u = x - e
    v = u / np.linalg.norm(u)
    Q_cnt = np.identity(r)
    Q_cnt[cnt:, cnt:] -= 2.0 * np.outer(v, v)
    R = np.dot(Q_cnt, R)  # R=H(n-1)*...*H(2)*H(1)*A
    Q = np.dot(Q, Q_cnt)  # Q = H (N-1) * ... * H (2) * H (1) H
```

```
np.set_printoptions(precision=4, suppress=True) #оставляем 4 знака после запятой
print(Q)
print(R)
```

```
[[ 0.647   0.5096  0.1799  0.5379]
 [ 0.5392 -0.4811  0.5743 -0.3848]
 [ 0.5392 -0.1305 -0.7902 -0.2607]
 [ 0.      0.7013  0.1162 -0.7034]]
[[ 9.2736  3.235  -5.3916]
 [ 0.      5.7039  2.006 ]
 [-0.      0.     13.7079]
 [ 0.      0.     -0.    ]]
```

Рисунок 5. Реализация метода Хаусхолдера

Метод вращений (метод Гивенса).

Метод вращений подробно изложен в курсе лекций, приведу только возможную реализацию метода на языке Python.

```
(r, c) = np.shape(A)
Q = np.identity(r)
R = np.copy(A)
(rows, cols) = np.tril_indices(r, -1, c)
for (row, col) in zip(rows, cols):
    if R[row, col] != 0:  # Q = 1, S = 0, R, Q без изменений
        r_ = np.hypot(R[col, col], R[row, col])  # d
        c = R[col, col]/r_
        s = -R[row, col]/r_
        G = np.identity(r)
        G[[col, row], [col, row]] = c
        G[row, col] = s
        G[col, row] = -s
        R = np.dot(G, R)  # R=G(n-1,n)*...*G(2n)*...*G(23,1n)*...*G(12)*A
        Q = np.dot(Q, G.T)  # Q=G(n-1,n).T*...*G(2n).T*...*G(23,1n).T*...*G(12).T
```

```
np.set_printoptions(precision=4, suppress=True) #оставляем 4 знака после запятой
print(Q)
print(R)
```

```
[[ 0.647   0.5096  0.1799 -0.5379]
 [ 0.5392 -0.4811  0.5743  0.3848]
 [ 0.5392 -0.1305 -0.7902  0.2607]
 [ 0.      0.7013  0.1162  0.7034]]
[[ 9.2736  3.235  -5.3916]
 [-0.      5.7039  2.006 ]
 [ 0.      0.     13.7079]
 [ 0.      0.      0.    ]]
```

Рисунок 6. Реализация метода Гивенса

Вопросы к лабораторной работе.

1. Верхнетреугольная, нижнетреугольная, унитреугольная матрицы.
2. Определение LU разложения.
3. Условие существования LU разложения.
4. Применение LU разложения.
5. Определение нормы.
6. Часто используемые нормы векторов и матриц
7. Определение ортогональной матрицы.
8. Определение унитарной матрицы.

9. Определение ортонормированного и ортогонального базисов.

10. Определение QR разложения.

11. Условие существования QR разложения.

12. Применение QR разложения.

Задание к лабораторной работе.

Вариант 1.

1. Создать квадратную матрицу из случайных целых чисел из [-3,3] размера 10. Найти и вычислить минор 4 порядка, расположенный на пересечении 2, 3, 4, 5 строк и 7, 8, 9, 10 столбцов. Использовать срезы матрицы.

2. Создать две матрицы из случайных вещественных чисел из интервала (2,3) подходящего размера. Найти их произведение тремя способами: используя векторный алгоритм умножения матриц; используя матричный алгоритм, записав матрицу C по столбцам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x10 из случайных целых чисел. Вычислить норму $\|x\|_1$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти спектральную норму матрицы с помощью самостоятельно написанного алгоритма (можно использовать функцию для нахождения собственных значений), проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 4 по 10. Например, вектор (1,2,3,4,5,6,7,8,9,10) переводит в вектор (1,2,c,0,0,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 2.

1. Создать квадратную матрицу из случайных вещественных чисел из (2,4) размера 10. Найти скалярное произведение 4 строки на 5 столбец. Использовать срезы матриц.

2. Создать две матрицы из случайных целых чисел из интервала [2,7) подходящего размера. Найти их произведение тремя способами: записав скалярный алгоритм умножения матриц; записав векторный алгоритм, записав матрицу С; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x10 из случайных целых чисел. Вычислить норму $\|x\|_2$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы $\|A\|_\infty$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 6 по 10. Например, вектор (1,2,3,4,5,6,7,8,9,10) переводит в вектор (1,2,3,4,c,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 3.

1. Создать квадратную матрицу из случайных целых чисел из [-5,2] размера 11. Найти и вычислить минор 5 порядка, расположенный на пересечении 1, 2, 3, 6, 7 строк и 7, 8, 9, 10, 11 столбцов. Использовать срезы матрицы.

2. Создать две матрицы из случайных вещественных чисел подходящего размера. Найти их произведение тремя способами: записав векторный алгоритм умножения матриц; записав матричный алгоритм, записав матрицу С по строкам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x8 из случайных целых чисел. Вычислить норму $\|x\|_\infty$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти фробениусову норму матрицы $\|A\|_F$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет все его координаты, кроме первой. Например, вектор (1,2,3,4,5,6,7,8) переводит в вектор (a,0,0,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 4.

1. Создать квадратную матрицу из случайных вещественных чисел из интервала (-1,1) размера 10. Найти скалярное произведение 2 строки на 7 столбец. Использовать срезы матриц.

2. Создать две матрицы из случайных целых чисел из [3,10] подходящего размера. Найти их произведение тремя способами: скалярный алгоритм умножения матриц; векторный алгоритм; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x10 из случайных целых чисел. Вычислить норму $\|x\|_\infty$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы $\|A\|_\infty$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 5 по 10. Например, вектор (1,2,3,4,5,6,7,8,9,10) переводит в вектор (1,2,3,d,0,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 5.

1. Создать квадратную матрицу из случайных целых чисел из [0,8] размера 6. Создать две новые матрицы: первая – из двух последних строк исходной матрицы (должна получиться матрица размера 2x6), вторая – из двух первых столбцов матрицы (матрица размера 6x2).

2. Выполнить умножение матриц из предыдущего пункта тремя способами: используя векторный алгоритм умножения матриц; используя матричный алгоритм, записав матрицу C по строкам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x10 из случайных целых чисел. Вычислить норму $\|x\|_3$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы Фробениуса $\|A\|_F$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 3 по 10. Например, вектор (1,2,3,4,5,6,7,8,9,10) переводит в вектор (1,b,0,0,0,0,0,0,0,0)

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 6.

1. Создать квадратную матрицу из случайных целых чисел из [-7,-2] размера 9. Найти и вычислить минор 4 порядка, расположенный на пересечении 1, 2, 3, 4 строк и 2, 3, 8, 9 столбцов. Использовать срезы матрицы.

2. Создать две матрицы из случайных вещественных чисел подходящего размера. Найти их произведение тремя способами: используя векторный алгоритм умножения матриц; используя матричный алгоритм, записав матрицу С по столбцам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x8 из случайных целых чисел. Вычислить норму $\|x\|_3$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти спектральную норму матрицы с помощью самостоятельно написанного алгоритма (можно использовать функцию для нахождения собственных значений), проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 4 по 8. Например, вектор (1,2,3,4,5,6,7,8) переводит в вектор (1,2,c,0,0,0,0,0)

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 7.

1. Создать квадратную матрицу из случайных вещественных чисел из (2,4) размера 8. Найти скалярное произведение 3 строки на 8 столбец. Использовать срезы матриц.

2. Создать две матрицы из случайных целых чисел из отрезка [-2,6] подходящего размера. Найти их произведение тремя способами: записав скалярный алгоритм умножения матриц; записав векторный алгоритм, записав матрицу С; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x9 из случайных целых чисел. Вычислить норму $\|x\|_2$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы $\|A\|_\infty$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в 3 пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 3 по 9. Например, вектор (1,2,3,4,5,6,7,8,9) переводит в вектор (1,b,0,0,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 8.

1. Создать квадратную матрицу из случайных целых чисел из [-4,3] размера 10. Найти и вычислить минор 5 порядка, расположенный на пересечении 1-5 строк и 2-4, 9, 10 столбцов. Использовать срезы матрицы.

2. Создать две матрицы из случайных вещественных чисел подходящего размера. Найти их произведение тремя способами: записав векторный алгоритм умножения матриц; записав матричный алгоритм, записав матрицу С по строкам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x5 из случайных целых чисел. Вычислить норму $\|x\|_5$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы $\|A\|_1$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет все его координаты, кроме первой. Например, вектор (1,2,3,4,5) переводит в вектор (a,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 9.

1. Создать квадратную матрицу из случайных вещественных чисел из интервала (-1,1) размера 8 . Найти скалярное произведение 1 строки на 8 столбец. Использовать срезы матриц.

2. Создать две матрицы из случайных целых чисел из [-6,6] подходящего размера. Найти их произведение тремя способами: скалярный алгоритм умножения матриц; векторный алгоритм; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x10 из случайных целых чисел. Вычислить норму $\|x\|_3$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы Фробениуса $\|A\|_F$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 3 по 10. Например, вектор (1,2,3,4,5,6,7,8,9,10) переводит в вектор (1,b,0,0,0,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

Вариант 10.

1. Создать квадратную матрицу из случайных целых чисел из [-5,5] размера 7. Создать две новые матрицы: первая – из трех последних строк исходной матрицы (должна получиться матрица размера 3x7), вторая – из двух первых столбцов матрицы (матрица размера 7x2).

2. Выполнить умножение матриц из предыдущего пункта тремя способами: используя векторный алгоритм умножения матриц; используя матричный алгоритм, записав матрицу C по строкам; проверив с помощью функции np.dot.

3. Создать вектор-строку 1x7 из случайных целых чисел. Вычислить норму $\|x\|_3$ самостоятельно написанной функцией и проверить результат с помощью linalg.norm().

4. Создать матрицу из случайных целых чисел. Найти норму матрицы Фробениуса $\|A\|_\infty$ с помощью самостоятельно написанного алгоритма, проверить результат с помощью linalg.norm().

5. Для вектора, созданного в третьем пункте, найти отражение Хаусхолдера, которое обнуляет его координаты с 3 по 7. Например, вектор (1,2,3,4,5,6,7) переводит в вектор (1,b,0,0,0,0,0).

6. Для матрицы, созданной в 4 пункте, найти LU разложение самостоятельно написанной функцией и проверить результат с помощью перемножения матриц.

7. Для матрицы, созданной в 4 пункте, найти QR разложение всеми рассмотренными методами, проверить результат с помощью перемножения матриц и с помощью функции np.linalg.qr.

3 РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

В линейной алгебре решаются следующие основные задачи.

1. Решение систем линейных уравнений.
2. Вычисление определителей.
3. Нахождение обратных матриц.
4. Определение собственных значений и собственных векторов.
5. Линейная задача метода наименьших квадратов.
6. Вычисление сингулярных чисел и сингулярных векторов.

Дана система линейных алгебраических уравнений

$$Ax = b \text{ , где}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mm} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{pmatrix}.$$

Будем предполагать, что матрица A задана и является невырожденной. Известно, что в этом случае решение системы существует, единственно и устойчиво по входным данным. Это означает, что рассматриваемая задача корректна.

Хотя задача решения системы сравнительно редко представляет самостоятельный интерес для приложений, от умения эффективно решать такие системы часто зависит сама возможность математического моделирования самых разнообразных процессов с применением компьютера. Как будет видно далее, значительная часть численных методов решения различных (в особенности нелинейных) задач включает в себя решение систем как элементарный шаг соответствующего алгоритма.

Пусть $x^*=(x_1^*, x_2^*, \ldots, x_m^*)^T$ – приближенное решение системы $Ax=b$. Мы будем стремиться к получению решения, для которого погрешность $e=x-x^*$ мала. Заметим, что качество полученного решения далеко не всегда характеризуется тем, насколько мала погрешность $e=x-x^*$. Иногда вполне удовлетворительным является критерий малости невязки $r=b-Ax^*$. Вектор r показывает, насколько отличается правая часть системы от левой, если подставить в нее приближенное решение. Заметим, что $r=Ax-Ax^*=A(x-x^*)$ и поэтому погрешность и невязка связаны равенством $e=x-x^*=A^{-1}r$.

3.1 Обусловленность задачи решения системы линейных алгебраических уравнений

Под обусловленностью вычислительной задачи понимают чувствительность ее решения к малым погрешностям входных данных. Задачу называют хорошо обусловленной, если малым погрешностям входных данных отвечают малые погрешности решения, и плохо обусловленной, если возможны сильные изменения решения. Часто оказывается возможным ввести количественную меру степени обусловленности вычислительной задачи – число обусловленности. Эту величину можно интерпретировать как коэффициент возможного возрастания погрешностей в решении по отношению к вызвавшим их погрешностям входных данных.

Задача вычисления решения x системы уравнений $Ax=b$ может быть как хорошо, так и плохо обусловленной. Пусть элементы матрицы *A* считаются заданными точно, а вектор-столбец правой части – приближенно.

Для погрешности приближенного решения системы $Ax=b$ справедлива оценка $\Delta(x^*)\leqslant\|A^{-1}\|\cdot\|r\|$, где $r=b-Ax^*$ – невязка, отвечающая x^*.

Пусть x^* – точное решение системы $Ax^*=b^*$, в которой правая часть b^* является приближением к b. Тогда верны следующие оценки абсолютной и относительной погрешностей:

$$\Delta(x^*)\leqslant \nu_\Delta \Delta(b^*),\ \ \delta(x^*)\leqslant \nu_\delta \delta(b^*),\ \text{где}\ \nu_\Delta=\|A^{-1}\|,$$

$$\nu_\delta(x)=\frac{\|A^{-1}\|\cdot\|b\|}{\|x\|}=\frac{\|A^{-1}\|\cdot\|Ax\|}{\|x\|}.$$

Величина $\nu_\Delta=\|A^{-1}\|$ для задачи $Ax=b$ играет роль абсолютного числа обусловленности.

Величина $\nu_\delta(x)=\dfrac{\|A^{-1}\|\cdot\|b\|}{\|x\|}$ называется естественным числом обусловленности.

Она зависит от конкретного решения x и характеризует коэффициент возможного возрастания относительной погрешности этого решения, вызванного погрешностью задания правой части. Это означает, что $\nu_\delta(x)$ для задачи вычисления решения x системы играет роль относительного числа обусловленности.

Вычислим максимальное значение естественного числа обусловленности

$$\max_{x\neq 0}\nu_\delta(x)=\max_{x\neq 0}\frac{\|A^{-1}\|\cdot\|Ax\|}{\|x\|}=\|A^{-1}\|\cdot\|A\|.$$

Полученную величину принято называть стандартным числом обусловленности (или просто числом обусловленности) матрицы A и обозначать через $\nu(A)$ или $cond(A)$.

Таким образом, $\nu(A)=cond(A)=\|A^{-1}\|\cdot\|A\|$.

Величина $cond(A)$ зависит, вообще говоря, от выбора нормы векторов в пространстве $\mathbb{R}^m$. Фактически это есть зависимость

максимального коэффициента роста погрешности от способа измерения величины входных данных и решения.

Пример 1. Вычислить $cond_\infty(A)$ для матрицы $A=\begin{pmatrix} 1{,}03 & 0{,}991 \\ 0{,}991 & 0{,}943 \end{pmatrix}$.

Решение. Сначала найдем обратную матрицу $A^{-1}\approx\begin{pmatrix} -87{,}4 & 91{,}8 \\ 91{,}8 & -95{,}4 \end{pmatrix}$.

Тогда $cond_\infty(A)=\|A^{-1}\|_\infty\cdot\|A\|_\infty\approx 187{,}2\cdot 2{,}021\approx 378$. Если входные данные для системы уравнений с матрицей A содержат относительную погрешность порядка 0,1-1%, то систему можно расценить как плохо обусловленную.

Рассмотрим систему уравнений $\begin{matrix} 1{,}03\,x_1 + 0{,}991\,x_2 = 2{,}51 \\ 0{,}991\,x_1 + 0{,}943\,x_2 = 2{,}41 \end{matrix}$.

Ее решением является $x_1\approx 1{,}981$, $x_2\approx 0{,}474$. Возмутим каждую из компонент вектора правой части $b=(2{,}51,\ 2{,}41)^T$ на *0,005*, $b^*=(2{,}515,\ 2{,}415)^T$ теперь решение $x^*_1\approx 2{,}877$, $x^*_2\approx -0{,}463$. Таким образом, решение оказалось полностью искаженным.

Относительная погрешность задания правой части $\delta(b^*)=\dfrac{\|b-b^*\|_\infty}{\|b\|_\infty}=\dfrac{0{,}005}{2{,}51}\approx 0{,}2\%$ привела к относительной погрешности решения $\delta(x^*)=\dfrac{\|x-x^*\|_\infty}{\|x\|_\infty}=\dfrac{0{,}9364}{1.981}\approx 47{,}3\%$.

Следовательно, погрешность возросла примерно в 237 раз.

Можно ли внести в правую часть системы такую погрешность, чтобы получить существенно большее, чем 237, значение коэффициента роста погрешности?

Вычислим естественное число обусловленности, являющееся максимальным значением рассматриваемого коэффициента, отвечающим решению $x \approx (1{,}981,\ 0{,}474)^T$ и

$$\nu_\delta(x) = \frac{\|A^{-1}\|_\infty \cdot \|b\|_\infty}{\|x\|_\infty} \approx \frac{187{,}2 \cdot 2{,}51}{1{,}981} \approx 237 .$$

Таким образом, на поставленный вопрос следует ответить отрицательно.

Можно дать следующую геометрическую интерпретацию рассмотренного примера. Каждому уравнению системы соответствует прямая на плоскости Ox_1x_2. По коэффициентам при x_1 и x_2 в этих уравнениях видно, что прямые почти параллельны. Так как вблизи точки пересечения прямые почти сливаются, то даже незначительная погрешность в задании положения этих прямых существенно меняет положение точки пересечения.

Традиционным примером очень плохо обусловленной матрицы является матрица Гильберта – матрица H с элементами $h_{ij} = \frac{1}{i + j - 1}$.

До сих пор мы предполагали, что матрица A задана точно. Однако на практике это часто не так. Пусть $x*$ – точное решение системы $A_* x^* = b$ с приближенно заданной матрицей A_*. Тогда верна следующая оценка относительной погрешности:

$$\delta^*(x^*) \leqslant cond(A) \cdot \delta(A_*) \text{, где } \delta^*(x^*) = \frac{\|x - x^*\|}{\|x^*\|},\ \delta(A_*) = \frac{\|A - A_*\|}{\|A\|} .$$

Проверить чувствительность решения системы $Ax = b$ к погрешностям можно и экспериментально. Для этого достаточно решить задачу несколько раз с близкими к b правыми частями

$b^{(1)}$, $b^{(2)}, \ldots, b^{(n)}$. Можно ожидать, что величина $\widetilde{\nu}_\delta = \max\limits_{1 \leqslant l \leqslant n} \dfrac{\delta(x^{(l)})}{\delta(b^{(l)})}$ даст оценку значения $\nu_\delta(x)$. Во всяком случае она дает оценку снизу, так как $\widetilde{\nu}_\delta \leqslant \nu_\delta(x) \leqslant cond(A)$.

3.2 Метод Гаусса. Схема единственного деления и *LU*-разложение

Вычисления с помощью метода Гаусса состоят из двух основных этапов, называемых прямым ходом и обратным ходом (обратной подстановкой). Прямой ход метода Гаусса заключается в последовательном исключении неизвестных из системы $Ax = b$ для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисления значений неизвестных производят на этапе обратного хода.

При выполнении вычислений 1-го шага исключения по схеме единственного деления система уравнений $Ax = b$ приводится к виду $A^{(1)}x = b^{(1)}$, где

$$A^{(1)} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1m} \\ 0 & a_{22}^{(1)} & \ldots & a_{2m}^{(1)} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & a_{m2}^{(1)} & \ldots & a_{mm}^{(1)} \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ \ldots \\ b_m^{(1)} \end{pmatrix},$$

где $a_{ij}^{(1)} = a_{ij} - \mu_{i1} a_{1j}$, $b^{(1)} = b_i - \mu_{i1} b_1$, $\mu_{i1} = \dfrac{a_{i1}}{a_{11}}$, $i, j = 2,3, \ldots, m$.

Введем матрицу

$$M_1 = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ -\mu_{21} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ -\mu_{m1} & 0 & \ldots & 1 \end{pmatrix},$$

справедливы равенства $A^{(1)}=M_1 A$, $b^{(1)}=M_1 b$, т. е. преобразование системы к виду $A^{(1)}x=b^{(1)}$ эквивалентно умножению левой и правой частей системы на матрицу M_1. Аналогично можно показать, что вычисления 2-го шага исключения приводят систему $A^{(2)}x=b^{(2)}$, где $A^{(2)}=M_2 A^{(1)}$, $b^{(2)}=M_2 b^{(1)}$,

$$A^{(2)}=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2m}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3m}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{m3}^{(2)} & \dots & a_{mm}^{(2)} \end{pmatrix},$$

$$M_1=\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & -\mu_{32} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & -\mu_{m2} & 0 & \dots & 1 \end{pmatrix}, \; b^{(2)}=\begin{pmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \dots \\ b_m^{(2)} \end{pmatrix}.$$

После (*m-1*)-го шага, завершающего прямой ход, система оказывается приведенной к виду $A^{(m-1)}x=b^{(m-1)}$ с верхней треугольной матрицей $A^{(m-1)}$. Здесь $A^{(m-1)}=M_{m-1}A^{(m-2)}$, $b^{(m-1)}=M_{m-1}b^{(m-2)}$, где

$$A^{(m-1)}=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2m}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3m}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{mm}^{(m-1)} \end{pmatrix},$$

$$M_{m-1}=\begin{pmatrix}1 & 0 & 0 & ... & 0 & 0\\ 0 & 1 & 0 & ... & 0 & 0\\ 0 & 0 & 1 & ... & 0 & 0\\ ... & ... & ... & ... & ... & ...\\ 0 & 0 & 0 & ... & -\mu_{m,m-1} & 1\end{pmatrix},\ b^{(m-1)}=\begin{pmatrix}b_1\\ b_2^{(1)}\\ b_3^{(2)}\\ ...\\ b_m^{(m-1)}\end{pmatrix}.$$

Матрица $A^{(m-1)}$ получена из матрицы A последовательным умножением на $M_1, M_2, ..., M_{m-1}$: $A^{(m-1)}=M_{m-1}...M_2M_1A$.

Аналогично, $b^{(m-1)}=M_{m-1}...M_2M_1b$. Следовательно,

$$A=M_1^{-1}M_2^{-1}...M_{m-1}^{-1}A^{(m-1)},\ M_1^{-1}=\begin{pmatrix}1 & 0 & 0 & ... & 0\\ \mu_{21} & 1 & 0 & ... & 0\\ \mu_{31} & 0 & 1 & ... & 0\\ ... & ... & ... & ... & ...\\ \mu_{m1} & 0 & 0 & ... & 1\end{pmatrix},$$

$$M_2^{-1}=\begin{pmatrix}1 & 0 & 0 & ... & 0\\ 0 & 1 & 0 & ... & 0\\ 0 & \mu_{32} & 1 & ... & 0\\ ... & ... & ... & ... & ...\\ 0 & \mu_{m2} & 0 & ... & 1\end{pmatrix},\ ...,\ M_{m-1}^{-1}=\begin{pmatrix}1 & 0 & 0 & ... & 0 & 0\\ 0 & 1 & 0 & ... & 0 & 0\\ ... & ... & ... & ... & ... & ...\\ 0 & 0 & 0 & ... & 1 & 0\\ 0 & 0 & 0 & ... & \mu_{m,m-1} & 1\end{pmatrix}.$$

Введем обозначения $U=A^{(m-1)}$, $L=M_1^{-1}M_2^{-1}\ldots M_{m-1}^{-1}$. Вычисляя матрицу L, убеждаемся в том, что она имеет следующий вид:

$$L=\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ \mu_{21} & 1 & 0 & \ldots & 0 \\ \mu_{31} & \mu_{32} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \mu_{m1} & \mu_{m2} & \mu_{m3} & \ldots & 1 \end{pmatrix}.$$

Тогда равенство $A=M_1^{-1}M_2^{-1}\ldots M_{m-1}^{-1}A^{(m-1)}$ в новых обозначениях примет вид $A=LU$. Это и есть $A=LU$ представление матрицы A в виде произведения нижней треугольной матрицы L и верхней треугольной матрицы U.

Если все главные миноры матрицы A отличны от нуля, то существуют единственные нижняя треугольная матрица L и верхняя треугольная матрица U такие, что $A=LU$.

В современных программах, реализующих метод Гаусса на компьютере, вычисления разбивают на два основных этапа. Первый этап это вычисление LU-разложения матрицы системы. Второй этап обработка правых частей и вычисление решения.

Смысл выделения первого этапа состоит в том, что он может быть выполнен независимо, для его проведения не нужна информация о правой части системы. Это как бы этап предварительной подготовки к быстрому вычислению решения. Именно для получения LU-разложения производится основная масса вычислений (примерно $(2/3)m^3$ арифметических операций).

На втором этапе выполняют следующие действия.

1. Преобразуют правую часть b по формулам прямого хода; необходимые для вычисления коэффициенты μ_{ij} берут из матрицы

L. В результате получают вектор $b^{(m-1)}$, связанный с вектором b формулой $b^{(m-1)} = M_{m-1} \ldots M_2 M_1 b$.

2. С помощью обратной подстановки решают систему $Ux = b^{(m-1)}$.

Для непосредственного вычисления решения x на втором этапе требуется примерно $2m^2$ арифметических операций.

В случае, если необходимо решить p систем уравнений с фиксированной матрицей A и различными правыми частями $d_{(1)}, d_{(2)}, \ldots, d_{(p)}$ первый этап проводят лишь один раз. Затем последовательно p раз проводят вычисления второго этапа для получения решений $x_{(1)}, x_{(2)}, \ldots, x_{(p)}$. Для этого требуется примерно $\frac{2}{3} m^2 + 2\,p\,m^2$ арифметических операций.

Пример 3. Методом Гаусса решить систему

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25 \\ 5x_1 + x_2 - 2x_3 + 4x_4 &= 14 \\ 3x_1 + 5x_2 + x_3 - x_4 &= 10 \\ 6x_2 - 2x_3 + 2x_4 &= 8. \end{aligned}$$

Решение. Прямой ход. 1-й шаг. Вычислим множители

$$\mu_{21} = \frac{a_{21}}{a_{11}} = \frac{5}{10} = 0{,}5\,, \quad \mu_{31} = \frac{a_{31}}{a_{11}} = \frac{3}{10} = 0{,}3\,, \quad \mu_{41} = \frac{a_{41}}{a_{11}} = \frac{0}{10} = 0\,.$$

Вычитая из второго, третьего и четвертого уравнений системы первое уравнение, умноженное на μ_{21}, μ_{31}, μ_{41} соответственно получаем:

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25 \\ -2x_2 - 3x_3 + 4x_4 &= 1{,}5 \\ 3{,}2x_2 + 0{,}4x_3 - x_4 &= 2{,}5 \\ 6x_2 - 2x_3 + 2x_4 &= 8. \end{aligned}$$

2-й шаг. Вычислим множители

$$\mu_{32} = \frac{a_{32}^{(1)}}{a_{22}^{(1)}} = \frac{3{,}2}{-2} = -1{,}6\ ,\ \ \mu_{42} = \frac{6}{-2} = -3\ .$$

Вычитая из третьего и четвертого уравнений системы второе уравнение, умноженное на μ_{32}, μ_{42} соответственно, приходим к системе

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25 \\ -2x_2 - 3x_3 + 4x_4 &= 1{,}5 \\ -4{,}4x_3 + 5{,}4x_4 &= 4{,}9 \\ -11x_3 + 14x_4 &= 12{,}5. \end{aligned}$$

3-й шаг. Вычисляя множитель

$$\mu_{43} = \frac{-11}{-4{,}4} = 2{,}5$$

и вычитая из четвертого уравнения системы третье уравнение, умноженное на μ_{43}, приводим систему к треугольному виду:

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 25 \\ -2x_2 - 3x_3 + 4x_4 &= 1{,}5 \\ -4{,}4x_3 + 5{,}4x_4 &= 4{,}9 \\ 0{,}5x_4 &= 0{,}25. \end{aligned}$$

Обратный ход. Из последнего уравнения системы находим $x_4 = 0{,}5$. Подставляя значение x_4 в третье уравнение, находим $x_3 = \dfrac{4{,}9 - 5{,}4\,x_4}{-4{,}4} = \dfrac{4{,}9 - 5{,}4 \cdot 0{,}5}{-4{,}4} = -0{,}5$.

Продолжая далее обратную подстановку, получаем:

$$x_2 = \frac{1{,}5 + 3\,x_3 - 4\,x_4}{-2} = 1\,,\quad x_1 = \frac{25 - 6\,x_2 - 2\,x_3}{10} = 2\,.$$

Итак, $x_1 = 2$, $x_2 = 1$, $x_3 = -0{,}5$, $x_4 = 0{,}5$.

Заметим, что вычисление множителей, а также обратная подстановка предполагают деление на главные элементы $a_{kk}^{(1)}$. Поэтому, если один из главных элементов оказывается равным нулю, то схема единственного деления не может быть реализована. Здравый смысл подсказывает, что и в ситуации, когда все главные элементы отличны от нуля, но среди них есть близкие к нулю, возможен неконтролируемый рост погрешности.

3.3 Метод Гаусса с выбором главного элемента и разложение матрицы на множители

В отличие от схемы единственного деления схема частичного выбора предполагает на k-м шаге прямого хода перестановку уравнений системы с номерами i_k и k (при выборе в качестве главного элемента k-го шага элемента $a_{i_k k}^{(k-1)}$). Это преобразование эквивалентно умножению системы на матрицу P_k, которая получается из единичной матрицы перестановкой i_k-й и k-й строк. Исключение неизвестного на k-м шаге по-прежнему эквивалентно умножению системы на матрицу M_k.

Таким образом, после 1-го шага система $Ax=b$ преобразуется к виду $A^{(1)}x=b^{(1)}$, где $A^{(1)}=M_1P_1A$, $b^{(1)}=M_1P_1b$. После 2-го шага система преобразуется к виду $A^{(2)}x=b^{(2)}$, где $A^{(2)}=M_2P_2A^{(1)}$, $b^{(2)}=M_2P_2b^{(1)}$. После завершающего (m-1)-го шага прямого хода система оказывается приведенной к виду $A^{(m-1)}x=b^{(m-1)}$, где $A^{(m-1)}=M_{m-1}P_{m-1}A^{(m-2)}$, $b^{(m-1)}=M_{m-1}P_{m-1}b^{(m-2)}$.

Как нетрудно видеть,

$$A^{(m-1)}=M_{m-1}P_{m-1}\ldots M_2P_2M_1P_1A,$$
$$b^{(m-1)}=M_{m-1}P_{m-1}\ldots M_2P_2M_1P_1b.$$

$$A=P_1^{-1}M_1^{-1}P_2^{-1}M_2^{-1}\ldots P_{m-1}^{-1}M_{m-1}^{-1}U,$$

где $U=A^{(m-1)}$ – верхняя треугольная матрица. Данное разложение не является *LU*-разложением матрицы *A*. Однако прямой ход по-прежнему равносилен *LU*-разложению, но уже не самой матрицы *A*, а матрицы $\widetilde{A}$, полученной из нее в результате соответствующей перестановки строк. Это разложение имеет вид $\widetilde{A}=\widetilde{L}U$, где $\widetilde{A}=P_{m-1}P_{m-2}\ldots P_2P_1A$, $\widetilde{L}$ нижняя треугольная матрица, отличающаяся от матрицы *L* перестановкой множителей в столбцах. После получения данного разложения для решения системы $Ax=b$ выполняют следующие действия.

1. Правую часть перестановкой элементов приводят к виду $\widetilde{b}=P_{m-1}P_{m-2}\ldots P_2P_1b$.

2. Преобразуют вектор $\widetilde{b}$ по формулам прямого хода; необходимые для вычислений множители $\widetilde{\mu}_{ij}$ берут из матрицы $\widetilde{L}$. В результате получают вектор $b^{(m-1)}$.

3. Обратной подстановкой решают систему $Ux=b^{(m-1)}$.

Заметим, что матрица перестановки P_k полностью определяется заданием номера i_k уравнения, которое переставляется с k-м уравнением. Поэтому для хранения всей информации о перестановках достаточно целочисленного массива длины m-1.

3.4 Метод Холецкого (метод квадратных корней)

Пусть требуется решить систему линейных алгебраических уравнений $Ax=b$ с симметричной положительно определенной матрицей A. Системы уравнений такого типа часто встречаются в приложениях (например, в задачах оптимизации, при решении уравнений математической физики и др.). Для их решения весьма часто применяется метод Холецкого.

В основе метода лежит алгоритм построения специального LU-разложения матрицы A, в результате чего она приводится к виду

$$A=L\,L^T\,.$$

В разложении Холецкого нижняя треугольная матрица

$$L=\begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{m1} & l_{m2} & l_{m3} & \dots & l_{mm} \end{pmatrix}$$

уже не обязательно должна иметь на главной диагонали единицы, как это было в методе Гаусса, а требуется только, чтобы диагональные элементы l_{ii} были положительными.

Если разложение $A=L\,L^T$ получено, то решение системы сводится к последовательному решению двух систем с треугольными матрицами: $Ly=b$, $L^T x=y$. Для решения этих систем требуется выполнение примерно $2m^2$ арифметических операций.

Найдем элементы матрицы L. Для этого вычислим элементы матрицы LL^T и приравняем их соответствующим элементам матрицы A. В результате получим систему уравнений:

$$
\begin{gathered}
l_{11}^2 = a_{11};\\
l_{i1} l_{11} = a_{i1}, \quad i = 2,3,\ldots m;\\
l_{21}^2 + l_{22}^2 = a_{22};\\
l_{i1} l_{21} + l_{i2} l_{22} = a_{i2}, \quad i = 3,4,\ldots m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
l_{k1}^2 + l_{k2}^2 + \ldots + l_{kk}^2 = a_{kk};\\
l_{i1} l_{k1} + l_{i2} l_{k2} + \ldots + l_{ik} l_{kk} = a_{ik}, \quad i = k+1,\ldots m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
l_{m1}^2 + l_{m2}^2 + \ldots + l_{mm}^2 = a_{mm}.
\end{gathered}
$$

Решая систему, последовательно находим

$$
\begin{gathered}
l_{11} = \sqrt{a_{11}};\\
l_{i1} = \frac{a_{i1}}{l_{11}}, \quad i = 2,3,\ldots m;\\
l_{22} = \sqrt{(a_{22} - l_{21}^2)};\\
l_{i2} = \frac{(a_{i2} - l_{i1} l_{21})}{l_{22}}, \quad i = 3,4,\ldots m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
l_{kk} = \sqrt{(a_{kk} - l_{k1}^2 - \ldots - l_{k,k-1}^2)};\\
l_{ik} = \frac{(a_{ik} - l_{i1} l_{k1} - l_{i2} l_{k2} - \ldots - l_{i,k-1} l_{k,k-1})}{l_{kk}}, \quad i = k+1,\ldots m;\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
l_{mm} = \sqrt{(a_{mm} - l_{m1}^2 - l_{m2}^2 - \ldots - l_{m,m-1}^2)}.
\end{gathered}
$$

Заметим, что для вычисления диагональных элементов используется операция извлечения квадратного корня. Поэтому метод Холецкого называют еще и методом квадратных корней. Дока-

зано, что положительность соответствующих подкоренных выражений является следствием положительной определенности матрицы A. Матрица L, входящая в разложение Холецкого определяется по матрице А однозначно.

Метод Холецкого при больших m требует вдвое меньше вычислительных затрат по сравнению с методом Гаусса. Учет симметричности матрицы A позволяет экономно использовать память компьютера при записи исходных данных задачи и результатов вычислений. Действительно, для задания матрицы A достаточно ввести в память компьютера только элементы a_{ij}, $i \leqslant j$, расположенные на главной диагонали и под ней. В формулах каждый такой элемент a_{ij} используется лишь однажды для получения l_{ij} и далее в вычислениях не участвует. Поэтому в процессе вычислений найденные элементы l_{ij} могут последовательно замещать элементы a_{ij}. В результате нижняя треугольная матрица L может быть расположена в той области памяти, где первоначально хранилась нижняя треугольная часть матрицы А. Применение для решения данной системы метода Гаусса потребовало бы использования примерно вдвое большего объема памяти. Безусловным достоинством метода Холецкого является также его гарантированная устойчивость.

Пример 4. Использовав метод Холецкого, решить систему

$$\begin{aligned} 6{,}25\,x_1 - x_2 + 0{,}5\,x_3 &= 7{,}5 \\ -x_1 + 5x_2 + 2{,}12\,x_3 &= -8{,}68 \\ 0{,}5\,x_1 + 2{,}12\,x_2 + 3{,}6\,x_3 &= -0{,}24. \end{aligned}$$

Решение. По формулам последовательно находим:

$$l_{11}=\sqrt{a_{11}}=\sqrt{6{,}25}=2{,}5\ ,\ l_{21}=\frac{a_{21}}{l_{11}}=\frac{-1}{2{,}5}=-0{,}4\ ,$$

$$l_{31}=\frac{a_{31}}{l_{11}}=\frac{0{,}5}{2{,}5}=0{,}2\ ,\ l_{22}=\sqrt{a_{22}-l_{21}^2}=\sqrt{5-0{,}16}=2{,}2\ ,$$

$$l_{32}=\frac{a_{32}-l_{31}l_{21}}{l_{22}}=\frac{2{,}12-0{,}2\cdot(-0{,}4)}{2{,}2}=1\ ,$$

$$l_{33}=\sqrt{a_{33}-l_{31}^2-l_{32}^2}=\sqrt{3{,}6-0{,}2^2-1^2}=1{,}6\ .$$

Следовательно, матрица L такова:

$$L=\begin{pmatrix}2{,}5 & 0 & 0\\ -0{,}4 & 2{,}2 & 0\\ 0{,}2 & 1 & 1{,}6\end{pmatrix}.$$

Система $Ly=b$ имеет вид

$$\begin{aligned}2{,}5\,y_1 &= 7{,}5\\ -0{,}4\,y_1+2{,}2\,y_2 &= -8{,}68\\ 0{,}2\,y_1+\ y_2+1{,}6\,y_3 &= -0{,}24.\end{aligned}$$

Решая ее, получаем $y_1=3$, $y_2=-3{,}4$, $y_3=1{,}6$.

Далее, из системы $L^Tx=y$, которая имеет вид

$$\begin{aligned}2{,}5\,x_1-0{,}4\,x_2+\ 0{,}2\,x_3 &= 3\\ 2{,}2\,x_2+x_3 &= -3{,}4\\ 1{,}6\,x_3 &= 1{,}6,\end{aligned}$$

находим решение $x_1=0{,}8$, $x_2=-2$, $x_3=1$.

3.5 Метод прогонки

Это простой и эффективный алгоритм решения систем линейных алгебраических уравнений с трехдиагональными матрицами

$$
\begin{aligned}
b_1 x_1 + c_1 x_2 &= d_1 \\
a_2 x_1 + b_2 x_2 + c_2 x_3 &= d_2 \\
&\cdots\cdots\cdots\cdots \\
a_i x_{i-1} + b_i x_i + c_i x_{i+1} &= d_i \\
&\cdots\cdots\cdots\cdots \\
a_{m-1} x_{m-2} + b_{m-1} x_{m-1} + c_{m-1} x_m &= d_{m-1} \\
a_m x_{m-1} + b_m x_m &= d_m
\end{aligned}
$$

Системы такого вида часто возникают при решении различных вычислительных задач (например, приближения функций сплайнами). Преобразуем первое уравнение системы к виду

$$x_1 = \alpha_1 x_2 + \beta_1 \text{, где } \alpha_1 = -\frac{c_1}{b_1},\ \beta_1 = \frac{d_1}{b_1}.$$

Подставим полученное для x_1 выражение во второе уравнение системы: $a_2(\alpha_1 x_2 + \beta_1) + b_2 x_2 + c_2 x_3 = d_2$. Преобразуем это уравнение к виду $x_2 = \alpha_2 x_3 + \beta_2$, где $\alpha_2 = -\dfrac{c_2}{b_2 + a_2 \alpha_1}$, $\beta_2 = \dfrac{d_2 - a_2 \beta_1}{b_2 + a_2 \alpha_1}$. Подставим полученное для x_2 выражение в третье уравнение системы и т. д.

На *i*-м шаге этого процесса $1 < i < m$ *i*-е уравнение системы преобразуется к виду $x_i = \alpha_i x_{i+1} + \beta_i$, где $\alpha_i = -\dfrac{c_i}{b_i + a_i \alpha_{i-1}}$, $\beta_i = \dfrac{d_i - a_i \beta_{i-1}}{b_i + a_i \alpha_{i-1}}$.

На *m*-м шаге подстановка в последнее уравнение выражения $x_{m-1} = \alpha_{m-1} x_m + \beta_{m-1}$ дает $a_m(\alpha_{m-1} x_m + \beta_{m-1}) + b_m x_m = d_m$.

Откуда можно определить значение $x_m=\beta_m=\dfrac{d_m-a_m\beta_{m-1}}{b_m+a_m\alpha_{m-1}}$. Значения остальных неизвестных x_i для $i=m-1,\ m-2,\ \ldots,\ 1$ теперь легко вычисляются по формуле $x_i=\alpha_i x_{i+1}+\beta_i$.

Сделанные преобразования позволяют организовать вычисления метода прогонки в два этапа.

Прямой ход метода прогонки (прямая прогонка) состоит в вычислении прогоночных коэффициентов $\alpha_i,\ \beta_i,\ 1\leqslant i<m$. При $i=1$ коэффициенты вычисляют по формулам

$$\alpha_1=-\frac{c_1}{\gamma_1},\ \beta_1=\frac{d_1}{\gamma_1},\ \gamma_1=b_1,$$

а при $i=2,\ 3,\ \ldots,\ m-1$ по рекуррентным формулам

$$\alpha_i=-\frac{c_i}{\gamma_i},\ \beta_i=\frac{d_i-a_i\beta_{i-1}}{\gamma_i},\ \gamma_i=b_i+a_i\alpha_{i-1}.$$

При $i=m$ прямая прогонка завершается вычислением

$$\beta_m=\frac{d_m-a_m\beta_{m-1}}{\gamma_m},\ \gamma_m=b_m+a_m\alpha_{m-1}.$$

Обратный ход метода прогонки (обратная прогонка), дает значения неизвестных. Сначала полагают $x_m=\beta_m$. Затем значения остальных неизвестных вычисляют по формуле

$$x_i=\alpha_i x_{i+1}+\beta_i,\ i=m-1,\ m-2,\ \ldots,\ 1.$$

Вычисления ведут в порядке убывания значений i.

Пример 5. Использовав метод прогонки, решить систему

$$\begin{aligned} 5x_1 - x_2 &= 2{,}0 \\ 2x_1 + 4{,}6x_2 - x_3 &= 3{,}3 \\ 2x_2 + 3{,}6x_3 - 0{,}8x_4 &= 2{,}6 \\ 3x_3 + 4{,}4x_4 &= 7{,}2 \end{aligned}.$$

Решение. Прямой ход. По формулам последовательно находим прогоночные коэффициенты.

$$\gamma_1 = b_1 = 5\,,\ \alpha_1 = \frac{-c_1}{\gamma_1} = \frac{1}{5} = 0{,}2\,,\ \beta_1 = \frac{d_1}{\gamma_1} = \frac{2{,}0}{5} = 0{,}4\,,$$

$$\gamma_2 = b_2 + a_2\alpha_1 = 4{,}6 + 2\cdot 0{,}2 = 5\,,\ \alpha_2 = \frac{-c_2}{\gamma_2} = \frac{1}{5} = 0{,}2\,,$$

$$\beta_2 = \frac{d_2 - a_2\beta_1}{\gamma_2} = \frac{3{,}3 - 2\cdot 0{,}4}{5} = 0{,}5\,,$$

$$\gamma_3 = b_3 + a_3\alpha_2 = 3{,}6 + 2\cdot 0{,}2 = 4\,,\ \alpha_3 = \frac{-c_3}{\gamma_3} = \frac{0{,}8}{4} = 0{,}2\,,$$

$$\beta_3 = \frac{d_3 - a_3\beta_2}{\gamma_3} = \frac{2{,}6 - 2\cdot 0{,}5}{4} = 0{,}4\,,$$

$$\gamma_4 = b_4 + a_4\alpha_3 = 4{,}4 + 3\cdot 0{,}2 = 5\,,\ \beta_4 = \frac{d_4 - a_4\beta_3}{\gamma_4} = \frac{7{,}2 - 3\cdot 0{,}4}{5} = 1{,}2\,.$$

Обратный ход. Полагаем $x_4 = \beta_4 = 1{,}2$. Далее находим:

$$x_3 = \alpha_3 x_4 + \beta_3 = 0{,}2\cdot 1{,}2 + 0{,}4 = 0{,}64\,,$$

$$x_2 = \alpha_2 x_3 + \beta_2 = 0{,}2\cdot 0{,}64 + 0{,}5 = 0{,}628\,,$$

$$x_1 = \alpha_1 x_2 + \beta_1 = 0{,}2\cdot 0{,}628 + 0{,}4 = 0{,}5256\,.$$

Итак, получаем решение:

$$x_1 = 0{,}5256\,,\ x_2 = 0{,}628\,,\ x_3 = 0{,}64\,,\ x_4 = 1{,}2\,.$$

Непосредственный подсчет показывает, что для реализации вычислений по данным формулам требуется примерно $8m$ арифметических операций, тогда как в методе Гаусса это число составляет примерно $(2/3)m^3$. Важно и то, что трехдиагональная структура матрицы системы позволяет использовать для ее хранения лишь $3m$-2 машинных слова. Таким образом, при одной и той же производительности и оперативной памяти компьютера метод прогонки позволяет решать системы гораздо большей размерности, чем стандартный метод Гаусса для систем уравнений с заполненной матрицей.

Приведем простые достаточные условия на коэффициенты системы, при выполнении которых вычисления по формулам прямой прогонки могут быть доведены до конца (ни один из знаменателей γ_i не обратится в нуль). В частности, это гарантирует существование решения системы и его единственность. При выполнении тех же условий коэффициенты α_i, при всех i удовлетворяют неравенству $|\alpha_i| \leqslant 1$, а следовательно, обратная прогонка устойчива по входным данным. Положим $a_1 = 0$, $c_m = 0$.

Пусть коэффициенты системы с трехдиагональными матрицами удовлетворяют следующим условиям диагонального преобладания: $|b_k| \geqslant |a_k| + |c_k|$, $|b_k| > |a_k|$, $1 \leqslant k \leqslant m$.

Тогда $\gamma_i \neq 0$ и $|\alpha_i| \leqslant 1$ для всех $i = 1, 2, \ldots, m$.

Описанный вариант метода прогонки можно рассматривать как одну из схем метода Гаусса (без выбора главного элемента), в результате прямого хода которого исходная трехдиагональная матрица

$$A=\begin{pmatrix} b_1 & c_1 & 0 & 0 & ... & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & ... & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & ... & 0 & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & 0 & ... & a_{m-1} & b_{m-1} & c_{m-1} \\ 0 & 0 & 0 & 0 & ... & 0 & a_m & b_m \end{pmatrix}$$

представляется в виде произведения двух двухдиагональных матриц: $A=LU$

$$L=\begin{pmatrix} \gamma_1 & 0 & 0 & ... & 0 & 0 \\ \alpha_2 & \gamma_2 & 0 & ... & 0 & 0 \\ 0 & \alpha_3 & \gamma_3 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \alpha_m & \gamma_m \end{pmatrix},$$

$$U=\begin{pmatrix} 1 & -\alpha_1 & 0 & ... & 0 & 0 \\ 0 & 1 & -\alpha_2 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 1 & -\alpha_{m-1} \\ 0 & 0 & 0 & ... & 0 & 1 \end{pmatrix}.$$

Так как для определения L и $A=LU$ нет необходимости вычислять коэффициенты β_i, то общее число операций на получение разложения составляет примерно $3m$. Данное разложение можно использовать для решения систем с многими правыми частями. Если нужно решить p систем с матрицей A, то общее число операций составит примерно $3m+5mp$. К сожалению, при обращении матрицы A теряется ее трехдиагональная структура. Обратная матрица является заполненной, однако для ее вычисления с помощью разложения LU требуется примерно $2{,}5m^2$ арифметических операций.

Определитель трехдиагональной матрицы, после того как получено разложение LU, вычисляется по элементарной формуле: $|A|=\gamma_1\gamma_2\ldots\gamma_m$.

Наряду со стандартным вариантом метода прогонки (правой прогонкой) существует большое число других вариантов этого метода. Это методы левой прогонки, встречных прогонок, немонотонной прогонки, потоковый вариант метода прогонки. В ряде случаев эти модификации могут существенно улучшить обусловленность прогонки. Для систем уравнений, обладающих близкой к

$$\begin{aligned}
&b_1x_1+c_1x_2 &&= d_1\\
&a_2x_1+b_2x_2+c_2x_3 &&= d_2\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_ix_{i-1}+b_ix_i+c_ix_{i+1} &&= d_i\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&a_{m-1}x_{m-2}+b_{m-1}x_{m-1}+c_{m-1}x_m &&= d_{m-1}\\
&a_mx_{m-1}+b_mx_m &&= d_m
\end{aligned}$$

структуре, разработаны методы циклической прогонки, матричной прогонки и др.

3.6 *QR* разложение матрицы

Рассмотрим два метода исключения, обладающих в отличие от метода Гаусса гарантированной хорошей обусловленностью – метод вращений и метод отражений. Оба этих метода позволяют получить представление исходной матрицы A в виде произведения ортогональной матрицы Q на верхнюю треугольную матрицу R: $A=QR$.

Вещественная матрица Q называется ортогональной, если для нее выполнено условие $Q^T=Q^{-1}$, что эквивалентно равенствам $Q^TQ=QQ^T=E$. Важное свойство ортогонального преобразования векторов (т. е. преобразования векторов с помощью их умножения

на ортогональную матрицу Q), состоит в том, что это преобразование не меняет евклидову норму векторов $\|Qx\|_2=\|x\|_2$ для всех $x \in \mathbb{R}^m$.

Метод вращений. Опишем прямой ход метода. На первом шаге неизвестное x_1 исключают из всех уравнений, кроме первого. Для исключения x_1 из второго уравнения вычисляют числа

$$c_{12}=\frac{a_{11}}{\sqrt{(a_{11}^2+a_{21}^2)}},\ s_{12}=\frac{a_{21}}{\sqrt{(a_{11}^2+a_{21}^2)}},$$

обладающие следующими свойствами:

$$c_{12}^2+s_{12}^2=1,\ -s_{12}a_{11}+c_{12}a_{21}=0.$$

Затем первое уравнение системы заменяют линейной комбинацией первого и второго уравнений с коэффициентами c_{12} и s_{12}, а второе уравнение – аналогичной линейной комбинацией с коэффициентам $-s_{12}$ и c_{12}.

В результате получают систему

$$\begin{aligned}
a_{11}^{(1)}x_1+a_{12}^{(1)}x_2+a_{13}^{(1)}x_3+\ldots+a_{1\mathrm{m}}^{(1)}x_m &= b_1^{(1)}\\
a_{22}^{(1)}x_2+a_{23}^{(1)}x_3+\ldots+a_{2\mathrm{m}}^{(1)}x_m &= b_2^{(1)}\\
a_{31}x_1+a_{32}x_2+a_{33}x_3+\ldots+a_{3\mathrm{m}}x_m &= b_3\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots &\ldots \quad \ldots\ldots\ldots\\
a_{m1}x_1+a_{m2}x_2+a_{m3}x_3+\ldots+a_{mm}x_m &= b_m,
\end{aligned}$$

в которой $a_{1j}^{(1)}=c_{12}a_{1j}+s_{12}a_{2j}$, $a_{2j}^{(1)}=-s_{12}a_{1j}+c_{12}a_{2j}$, $1\leqslant j\leqslant m$, $b_1^{(1)}=c_{12}b_1+s_{12}b_2$, $b_2^{(1)}=-s_{12}b_1+c_{12}b_2$.

Заметим, что $a_{21}^{(1)}=-s_{12}a_{11}+c_{12}a_{21}=0$ в силу специального выбора чисел c_{12} и s_{12}. Естественно, что если $a_{21}=0$, то в

исключении неизвестного x_1 из второго уравнения нет необходимости, полагают $c_{12}=1$ и $s_{12}=0$.

Как нетрудно видеть, преобразование исходной системы $Ax=b$ эквивалентно умножению слева матрицы A и правой части b на матрицу T_{12}, имеющую вид

$$T_{12}=\begin{pmatrix} c_{12} & s_{12} & 0 & 0 & \dots & 0 \\ -s_{12} & c_{12} & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}.$$

Для исключения x_1 из третьего уравнения вычисляют числа

$$c_{13}=\frac{a_{11}^{(1)}}{\sqrt{\left(\left(a_{11}^{(1)}\right)^2+a_{31}^2\right)}},\quad s_{13}=\frac{a_{31}}{\sqrt{\left(\left(a_{11}^{(1)}\right)^2+a_{31}^2\right)}}\ \text{такие, что}\ c_{13}^2+s_{13}^2=1,$$

$$-s_{13}a_{11}^{(1)}+c_{13}a_{31}=0.$$

Затем первое уравнение системы заменяют линейной комбинацией первого и второго уравнений с коэффициентами c_{13} и s_{13}, а третье уравнение – аналогичной линейной комбинацией с коэффициентами $-s_{13}$ и c_{13}. Это преобразование системы эквивалентно умножению слева на матрицу

$$T_{13}=\begin{pmatrix} c_{13} & 0 & s_{13} & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ -s_{13} & 0 & c_{13} & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

и приводит к тому, что коэффициент при x_1 в преобразованном третьем уравнении обращается в нуль. Таким же образом x_1 исключают из уравнений с номерами $i=4,\dots,m$.

В результате 1-го шага, состоящего из $m-1$ малых шагов, система приводится к виду

$$\begin{aligned} a_{11}^{(m-1)}x_1+a_{12}^{(m-1)}x_2+a_{13}^{(m-1)}x_3+\dots+a_{1\mathrm{m}}^{(m-1)}x_m &= b_1^{(m-1)} \\ a_{22}^{(1)}x_2+a_{23}^{(1)}x_3+\dots+a_{2\mathrm{m}}^{(1)}x_m &= b_2^{(1)} \\ a_{32}^{(1)}x_2+a_{33}^{(1)}x_3+\dots+a_{3\mathrm{m}}^{(1)}x_m &= b_3^{(1)} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots & \dots \quad \dots\dots\dots \\ a_{m2}^{(1)}x_2+a_{m3}^{(1)}x_3+\dots+a_{mm}^{(1)}x_m &= b_m^{(1)}. \end{aligned}$$

В матричной записи получаем:

$$A^{(1)}x=b^{(1)},$$

где $A^{(1)}=T_{1m}\dots T_{13}T_{12}A$, $b^{(1)}=T_{1m}\dots T_{13}T_{12}b$, T_{kl} – матрица элементарного преобразования, отличающаяся от единичной матрицы E только четырьмя элементами. В ней элементы с индексами (k,k) и (l,l) равны c_{kl}, элемент с индексами (k,l) равен s_{kl}, а элемент с индексами (l,k) равен $-s_{kl}$ причем выполнено условие $c_{kl}^2+s_{kl}^2=1$.

Действие матрицы T_{kl} на вектор x эквивалентно его повороту вокруг оси, перпендикулярной плоскости Ox_kx_l на угол ϕ_{kl} такой, что $c_{kl}=\cos\phi_{kl}$, $s_{kl}=\sin\phi_{kl}$ существование такого угла гарантируется равенством $c_{kl}^2+s_{kl}^2=1$. Эта геометрическая интерпретация и дала название методу вращений. Операцию умножения на матрицу T_{kl} часто называют плоским вращением (или преобразованием Гивенса). Заметим, что $T_{kl}^T=T_{kl}^{(-1)}$ и, следовательно, матрица T_{kl} ортогональная.

На 2-м шаге метода вращений, состоящем из $m-2$ малых шагов, из уравнений системы $A^{(1)}x=b^{(1)}$ с номерами $i=3,\ 4,\ \ldots,\ m$ исключают неизвестное x_2. Для этого каждое *i*-е уравнение комбинируют со вторым уравнением. В результате приходим к системе

$$A^{(2)}x=b^{(2)},\ \text{где}$$

$$A^{(2)}=T_{2m}\ldots T_{24}T_{23}A^{(1)},\ \ b^{(2)}=T_{2m}\ldots T_{24}T_{23}b^{(1)}.$$

После завершения *m*-1-го шага система принимает вид

$$\begin{aligned}
a_{11}^{(m-1)}x_1+a_{12}^{(m-1)}x_2+a_{13}^{(m-1)}x_3+\ldots+a_{1m}^{(m-1)}x_m &= b_1^{(m-1)}\\
a_{22}^{(m-1)}x_2+a_{23}^{(m-1)}x_3+\ldots+a_{2m}^{(m-1)}x_m &= b_2^{(m-1)}\\
a_{33}^{(m-1)}x_3+\ldots+a_{3m}^{(m-1)}x_m &= b_3^{(m-1)}\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots &\ldots\ \ldots\ldots\\
a_{mm}^{(m-1)}x_m &= b_m^{(m-1)}
\end{aligned}$$

или в матричной форме записи $A^{(m-1)}x=b^{(m-1)}$, где

$$A^{(m-1)}=T_{m-1,\,m}A^{(m-2)},\ \ b^{(m-1)}=T_{m-1,\,m}b^{(m-2)}.$$

Введем обозначение *R* для полученной верхней треугольной матрицы $A^{(m-1)}$. Она связана с исходной матрицей *A* равенством $R=TA$, где *T* – матрица результирующего вращения $T=T_{m-1,\,m}\ldots T_{2m}\ldots T_{23}T_{1\,m}\ldots.T_{13}T_{12}$. Заметим, что матрица *T* ортогональна как произведение ортогональных матриц. Обозначая $Q=T^{-1}=T^T$, получаем *QR*-разложение матрицы *A*.

Обратный ход метода вращений проводится точно так же, как и для метода Гаусса.

Метод вращений обладает замечательной численной устойчивостью. Однако этот метод существенно более трудоемок по сравнению с методом Гаусса. Получение *QR*-разложения для

квадратной матрицы A общего вида требует примерно $2m^3$ арифметических операций.

Пример 6. Использовав метод вращений, решить систему

$$\begin{aligned} 2x_1 - 9x_2 + 5x_3 &= -4 \\ 1{,}2x_1 - 5{,}3999x_2 + 6x_3 &= 0{,}6001 \\ x_1 - x_2 - 7{,}5x_3 &= -8{,}5 \end{aligned}.$$

Решение. Прямой ход. Для исключения x_1 из второго уравнения вычислим числа

$$c_{12} = \frac{a_{11}}{\sqrt{(a_{11}^2 + a_{21}^2)}} = \frac{2}{\sqrt{(2^2 + 1{,}2^2)}} \approx 0{,}857493 ,$$

$$s_{12} = \frac{a_{21}}{\sqrt{(a_{11}^2 + a_{21}^2)}} = \frac{1{,}2}{\sqrt{(2^2 + 1{,}2^2)}} \approx 0{,}514495 ,$$

Преобразуя коэффициенты первого и второго уравнений, приходим к системе

$$\begin{aligned} 2{,}33238x_1 - 10{,}4957x_2 + 7{,}37444x_3 &= -3{,}12122 \\ 7{,}85493 \cdot 10^{-5} x_2 + 2{,}57248x_3 &= 2{,}57256 \\ x_1 - x_2 - 7{,}5x_3 &= -8{,}5 \end{aligned}$$

Далее вычислим коэффициенты

$$c_{13} = \frac{a_{11}^{(1)}}{\sqrt{((a_{11}^{(1)})^2 + a_{31}^2)}} = \frac{2{,}33238}{\sqrt{2{,}33238^2 + 1^2}} \approx 0{,}919087 ,$$

$$s_{13} = \frac{a_{31}}{\sqrt{((a_{11}^{(1)})^2 + a_{31}^2)}} = \frac{1}{\sqrt{2{,}33238^2 + 1^2}} \approx 0{,}394055 .$$

Заменяя первое и третье уравнения их линейными комбинациями с коэффициентами c_{13}, s_{13}, $-s_{13}$, c_{13} соответственно, получаем систему

$$\begin{aligned} 2{,}53772\,x_1 - 10{,}0405\,x_2 + 3{,}82234\,x_3 &= -6{,}21814 \\ 7{,}85493\cdot 10^{-5}\,x_2 + 2{,}57248\,x_3 &= \ \ 2{,}57256 \\ 3{,}21680\,x_2 - 9{,}79909\,x_3 &= -6{,}58231\,. \end{aligned}$$

2-й шаг. В полученной системе имеем $a_{22}^{(1)} = 7{,}85493\cdot 10^{-5}$, $a_{32}^{(1)} = 3{,}21608$. Поэтому

$$c_{23} = \frac{a_{22}^{(1)}}{\sqrt{\left(\left(a_{22}^{(1)}\right)^2 + a_{32}^2\right)}} \approx 2{,}44185\cdot 10^{-5},\ s_{23} \approx 1{,}00000\ .$$

Заменяя второе и третье уравнения системы их линейными комбинациями с коэффициентами $c_{23},\ s_{23},\ -s_{23},\ c_{23}$ соответственно, приходим к системе

$$\begin{aligned} 2{,}53772\,x_1 - 10{,}0405\,x_2 + 3{,}82234\,x_3 &= -6{,}21814 \\ 3{,}21680\,x_2 + 9{,}79903\,x_3 &= -6{,}58225 \\ -2{,}57272\,x_3 &= -2{,}57272\,. \end{aligned}$$

Обратный ход дает последовательно значения

$$x_3 = 1,\ x_2 = 0{,}999994,\ x_1 = -1{,}58579\cdot 10^{-5}\,.$$

Метод отражений. Матрицами Хаусхолдера (или отражений) называются квадратные матрицы вида

$$V = E - 2\,w\,w^T\,,$$

где w – вектор-столбец в $\mathbb{R}^m$, имеющий единичную длину: $\|w\|_2 = 1$.

Матрица Хаусхолдера симметрична и ортогональна. Действительно,

$$V^T = \left(E - 2\,w\,w^T\right)^T = E^T - 2\left(w^T\right)^T w^T = E - 2\,w\,w^T = V\ ,$$

$$V^T V = VV = \left(E - 2\,w\,w^T\right)\left(E - 2\,w\,w^T\right) = E - 4\,w\,w^T + 4\,w\,w^T\,w\,w^T = E\ .$$

Учли, что $w w^T = \|w\|_2^2 = 1$.

Умножение на матрицу V называют преобразованием Хаусхолдера (или отражением). Действие V на вектор x можно интерпретировать как ортогональное отражение вектора в $\mathbb{R}^m$ относительно гиперплоскости, проходящей через начало координат и имеющей нормальный вектор, равный w.

Как и вращения, отражения используются для обращения в нуль элементов преобразуемой матрицы. Однако здесь с помощью одного отражения можно обратить в нуль уже не один элемент матрицы, а целую группу элементов некоторого столбца или строки. Поэтому, являясь почти столь же устойчивым, как и метод вращений, метод отражений позволяет получить QR-разложение квадратной матрицы общего вида примерно за $\frac{4}{3}m^3$ арифметических операций, т. е. в полтора раза быстрее.

Поясним, как получается QR-разложение матрицы A с помощью преобразований Хаусхолдера. Пусть $a = (a_1, a_2, \ldots a_m)^T$ произвольный вектор, у которого одна из координат отлична от нуля. Вектор w в преобразовании Хаусхолдера выбираем так, чтобы обратились в нуль все координаты вектора Va кроме первой: $Va = ce_1 = c(1,0,\ldots 0)^T$.

Поскольку ортогональное преобразование не меняет евклидову норму вектора, то $c = \|a\|_2$ и искомое преобразование таково, что $Va = (E - 2ww^T)a = a - 2(w,a)w = a - \alpha w = \|a\|_2 e_1$,

где $\alpha = 2(w,a)$. Таким образом, вектор w следует выбрать так: $w = \frac{v}{\|v\|_2}$, где $v = a \pm \|a\|_2 e_1$. Взяв $a = a_1$, где a_1 первый из столбцов матрицы A, и положив $P_1 = V$ получим

$$A^{(1)} = P_1 A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1m}^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2m}^{(1)} \\ \dots & \dots & \dots & \dots \\ 0 & a_{m2}^{(1)} & \dots & a_{mm}^{(1)} \end{pmatrix}.$$

Далее, взяв вектор $a_2 = (a_{22}^{(1)}, \dots a_{m2}^{(1)})^T \in \mathbb{R}^{m-1}$, с помощью преобразования Хаусхолдера V_{m-1} в пространстве (m-1)-мерных векторов можно обнулить все координаты вектора $V_{m-1} a_2$ кроме первой. Положив

$$P_2 = \begin{pmatrix} 1 & 0 \\ 0 & V_{m-1} \end{pmatrix}$$

получим:

$$A^{(2)} = P_2 A^{(1)} = P_2 P_1 A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1m}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2m}^{(2)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{2m}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{m3}^{(2)} & \dots & a_{mm}^{(2)} \end{pmatrix}.$$

Заметим, что первый столбец и первая строка матрицы при этом преобразовании не меняются. Выполнив m-1 шаг этого метода, приходим к верхней треугольной матрице

$$A^{(m-1)} = P_{m-1} \dots P_2 P_1 A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1m}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2m}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \dots & a_{2m}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{mm}^{(m-1)} \end{pmatrix}.$$

Поскольку матрицы $P_1 P_2 \dots P_{m-1}$ симметричны и ортогональны, то получено разложение $A = QR$, где матрица $Q = P_1 P_2 \dots P_{m-1}$ – ортогональная, а матрица $R = A^{(m-1)}$ – верхняя треугольная.

Пример 7. Использовав метод отражений, решить систему

$$\begin{aligned} 2x_1 - 9x_2 + 5x_3 &= -4 \\ 1{,}2x_1 - 5{,}3999x_2 + 6x_3 &= 0{,}6001 \\ x_1 - x_2 - 7{,}5x_3 &= -8{,}5. \end{aligned}$$

Решение. Прямой ход. Возьмем первый столбец матрицы системы $a_1 = (2,\ 1{,}2,\ 1)^T$, положим

$v = a_1 + \|a_1\|_2 e_1 = (2 + \sqrt{6{,}44},\ 1{,}2,\ 1)^T \approx (4{,}53772,\ 1{,}2,\ 1)^T$ и выберем

$$w = \frac{v}{\|v\|_2} \approx (0{,}945545,\ 0{,}250049,\ 0{,}208375)^T.$$

Построим матрицу Хаусхолдера

$$V = E - 2ww^T =$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 0{,}945545 \\ 0{,}250049 \\ 0{,}208375 \end{pmatrix} (0{,}945545 \quad 0{,}250049 \quad 0{,}208375) \approx$$

$$\approx \begin{pmatrix} -0{,}788110 & -0{,}472866 & -0{,}394056 \\ 0{,}472866 & 0{,}874951 & -0{,}104208 \\ -0{,}394056 & -0{,}104208 & 0{,}913160 \end{pmatrix}.$$

Применим к левой и правой частям системы преобразование Хаусхолдера и получим эквивалентную систему

$$\begin{aligned} -2{,}53772x_1 + 10{,}0405x_2 + 3{,}82233x_3 &= 6{,}21815 \\ -0{,}364646x_2 + 3{,}66694x_3 &= 3{,}30229 \\ 3{,}19606x_2 - 9{,}44423x_3 &= -6{,}24817. \end{aligned}$$

Возьмем теперь $a_2=(-0{,}364646,\ 3{,}19606)^T$ и построим двумерное преобразование Хаусхолдера. Здесь уже

$$v=a_2+\|a_2\|_2 e_1=(-0{,}364646+\sqrt{0{,}364646^2+3{,}19606^2},\ 3{,}19606)^T\approx$$
$$\approx(2{,}85215,\ 3{,}19606)^T$$

$$w=\frac{v}{\|v\|_2}\approx(\ 0{,}665824,\ 0{,}746109)^T\ .$$

Двумерная матрица Хаусхолдера имеет вид

$$V=E-2\mathrm{w}\,w^T=$$
$$=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}-2\begin{pmatrix}0{,}665824\\ 0{,}746109\end{pmatrix}(0{,}665824\quad 0{,}746109)\approx$$
$$\approx\begin{pmatrix}0{,}113357 & -0{,}99355\\ -0{,}99355 & -0{,}113357\end{pmatrix}.$$

Умножив обе части системы на матрицу

$$P_2=\begin{pmatrix}1 & 0 & 0\\ 0 & 0{,}113357 & -0{,}993555\\ 0 & -0{,}993555 & -0{,}113357\end{pmatrix},$$

получим

$$\begin{aligned}-2{,}53772\,x_1+10{,}0405\,x_2+3{,}82233\,x_3 &= 6{,}21815\\ -0{,}321608\,x_2+9{,}79904\,x_3 &= 6{,}58224\\ -2{,}57274\,x_3 &= -2{,}57273.\end{aligned}$$

Обратный ход последовательно дает значения

$$x_3\approx 0{,}999996,\ x_2\approx 0{,}999988,\ x_1\approx -3{,}35721\cdot 10^{-5}\ .$$

3.7 Линейная задача метода наименьших квадратов

Метод наименьших квадратов, в плане вычислений сводится к следующей ключевой задаче, которую принято называть линейной задачей метода наименьших квадратов. Пусть A – прямоугольная матрица размера $n \times m$ с элементами a_{ij}, $b \in \mathbb{R}^n$ заданный вектор. Требуется найти вектор $\bar{x} \in \mathbb{R}^m$, минимизирующий величину $\|Ax-b\|_2$, т. е. такой вектор $\bar{x}$, что $\|A\bar{x}-b\|_2 = \min\limits_{x \in \mathbb{R}^m} \|Ax-b\|_2$.

Линейная задача метода наименьших квадратов естественным образом возникает и при решении систем линейных алгебраических уравнений. Предположим, что требуется решить систему n линейных алгебраических уравнений с m неизвестными, где $n > m$

$$
\begin{array}{rcl}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1m}x_m & = & b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2m}x_m & = & b_2 \\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots & \ldots\ldots \\
a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nm}x_m & = & b_n.
\end{array}
$$

Системы уравнений, в которых число уравнений превышает число неизвестных, называют переопределенными. Для переопределенной системы, вообще говоря, нельзя найти вектор x, точно удовлетворяющий уравнениям системы. Однако, хотя уравнения системы нельзя удовлетворить точно, можно попытаться удовлетворить их как можно точнее, минимизируя величину вектора невязки $r = b - Ax$. Выбор в качестве минимизируемой величины евклидовой нормы невязки $\|r\|_2 = \|b - Ax\|_2$ приводит к методу наименьших квадратов решения переопределенных систем. Другими словами, в этом методе предлагается за решение переопределенной системы принять решение $\bar{x}$ задачи метода наименьших квадратов, обладающее свойством

$$\|A\bar{x}-b\|_2 = \min_{x \in \mathbb{R}^m} \|Ax-b\|_2 .$$

Простейшим подходом к решению линейной задачи метода наименьших квадратов является использование нормальной системы метода наименьших квадратов

$$A^T A x = A^T b .$$

Линейную систему называют нормальной если:

1) матрица коэффициентов A - симметрическая, т. е. $a_{ij} = a_{ji}$;

2) соответствующая квадратичная форма $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j$ положительно определенная.

Вектор $\bar{x}$ минимизирует величину $\|Ax - b\|_2$ тогда и только тогда, когда он является решением нормальной системы $A^T A x = A^T b$.

Столбцы матрицы A линейно зависимы тогда и только тогда, когда существует ненулевой вектор x такой, что $Ax = 0$.

Матрица $A^T A$ – симметричная, причем $(A^T A x, x) \geqslant 0$, $\forall x \in \mathbb{R}^m$.

Если столбцы матрицы A линейно независимы, то матрица $A^T A$ положительно определенная.

Линейная задача метода наименьших квадратов имеет решение. Это решение единственно тогда и только тогда, когда столбцы матрицы A линейно независимы.

Когда столбцы матрицы A линейно независимы, матрица $A^T A$ - симметричная и положительно определенная. Поэтому решение нормальной системы можно найти методом Холецкого примерно за $2n^2$ арифметических операций. Правда, для формирования системы (с учетом симметричности матрицы $A^T A$) нужно примерно $n^2 m$ арифметических операций.

3.8 Алгоритм Грама-Шмидта и *QR*-разложение матрицы

В случае, когда A – прямоугольная матрица размера $n \times m$, будем называть QR разложением матрицы A ее представление в виде $A = QR$, где Q – прямоугольная матрица размера $n \times m$ с ортонормированными столбцами $q_1, q_2, \ldots, q_m$, а R – верхняя треугольная квадратная матрица размера $m \times m$ с элементами r_{ij}.

Ортонормированность столбцов матрицы Q означает, что $(q_i, q_j) = 0$ для всех $i \neq j$ и $(q_j, q_j) = 1$ для всех j, что эквивалентно выполнению равенства $Q^T Q = E$, где E – единичная матрица порядка m.

Равенство $A = QR$ означает, что существуют векторы $q_1, q_2, \ldots, q_m \in \mathbb{R}^m$ и коэффициенты r_{ik} $1 \leqslant i \leqslant k \leqslant m$ такие, что для всех $k = 1,2,\ldots,m$ столбцы a_k матрицы A могут быть представлены в виде линейной комбинации

$$a_k = \sum_{i=1}^{k} r_{ik} q_i .$$

Применим алгоритм Грама-Шмидта для ортогонализации столбцов $a_1, a_2, \ldots, a_m$ матрицы A, предполагая, что они линейно независимы. Данный алгоритм состоит из m шагов, на каждом из которых вычисляются очередной вектор q_k и коэффициенты r_{ik} $1 \leqslant i \leqslant k$.

На первом шаге полагают $r_{11} = \|a_1\|_2$ и $q_1 = \dfrac{a_1}{r_{11}}$.

Пусть на (k-1)-м шаге найдена ортонормированная система векторов $q_1, q_2, \ldots, q_{k-1}$. Тогда на k-м шаге производят следующие вычисления:

$$r_{ik}=(a_k, q_i),\ i=1,2,\dots,k-1,\ \widetilde{q}=a_k-\sum_{i=1}^{k-1} r_{ik} q_i,\ r_{kk}=\|\widetilde{q}_k\|_2,$$

$$q_k=\frac{\widetilde{q}_k}{r_{kk}}.$$

Заметим, что $\widetilde{q}\neq 0$ и $r_{kk}=\|\widetilde{q}_k\|_2\neq 0$, так как в противном случае $a_k=\sum_{i=1}^{k-1} r_{ik} q_i$.

Так как каждый из векторов q_i, $1\leqslant i<k$ является линейной комбинацией векторов a_1, a_2, …, a_{k-1}, то равенство $a_k=\sum_{i=1}^{k-1} r_{ik} q_i$ означает, что вектор a_k является линейной комбинацией векторов a_1, a_2, …, a_{k-1}, что противоречит предположению о том, что столбцы матрицы *A* линейно независимы.

Обратим внимание на то, что $\|q_k\|_2=1$ и $(q_i, q_j)=0$ для всех $j=1,2,\dots,k-1$. Действительно

$$r_{kk}(q_k, q_j)=(\widetilde{q_k}, q_j)=(a_k, q_j)-\sum_{i=1}^{k-1} r_{ik}(q_i, q_j)=r_{jk}-r_{jk}=0.$$

Таким образом, полученная на *k*-м шаге алгоритма система векторов, q_1, q_2, …, q_k также является ортонормированной.

Пусть *A* – матрица размера $n\times m$, столбцы которой линейно независимы. Тогда существуют единственная матрица *Q* размера $n\times m$ с ортонормированными столбцами и верхняя треугольная матрица *R* с положительными диагональными элементами r_{ii} $1\leqslant i\leqslant m$ такие, что справедливо представление $A=QR$.

3.9 Модифицированный алгоритм Грама-Шмидта

К сожалению, классический алгоритм Грама-Шмидта обладает чрезмерно высокой чувствительностью к вычислительной погрешности. Поэтому для численной ортогонализации столбцов матрицы A используется модифицированный алгоритм Грама-Шмидта, устойчивый к вычислительной погрешности.

При отсутствии вычислительной погрешности этот алгоритм дает те же матрицы Q и R, что и классический алгоритм.

На k-м шаге этого алгоритма получают очередной столбец q_k матрицы Q, модифицированные столбцы $a_{k+1}^{(k)}, \ldots, a_m^{(k)}$ матрицы A и коэффициенты r_{kj} для $j = k,\ k+1,\ m$.

На первом шаге полагают $r_{11} = \|a_1\|_2$, $q_1 = \dfrac{a_1}{r_{11}}$, $a_j^{(1)} = a_j$, $j = 2, \ldots, m$.

Пусть на (k-1)-м шаге получены столбцы $q_1, q_2, \ldots, q_{k-1}$ и модифицированные столбцы $a_k^{(k-1)}, a_{k+1}^{(k-1)}, \ldots, a_m^{(k-1)}$. Тогда на k-м шаге производят следующие вычисления:

$$r_{kk} = \|a_k^{(k-1)}\|_2,\ q_k = \frac{a_k^{(k-1)}}{r_{kk}},$$

$$r_{kj} = (a_j^{(k)}, q_k),\ j = k+1, \ldots, m,$$

$$a_j^{(k)} = a_j^{(k-1)} - r_{kj} q_k,\ j = k+1, \ldots, m.$$

Для получения QR разложения матрицы A могут быть применены и другие методы.

Для решения линейной задачи метода наименьших квадратов применяют QR разложение. Пусть QR разложение матрицы A найдено. Вектор $\bar{x}$ минимизирует величину $\|Ax - b\|_2$ тогда и только тогда, когда он является решением нормальной системы

$A^T A x = A^T b$. Вектор $\bar{x}$, являющийся решением линейной задачи метода наименьших квадратов, удовлетворяет нормальной системе. Используя разложения

$$A = QR \text{ и } A^T A = (QR)^T QR = R^T Q^T Q R = R^T R$$

запишем нормальную систему в виде:

$$R^T Rx = R^T Q^T b .$$

Положив $c = Q^T b$ и умножив левую и правую части этой системы на $(R^T)^{-1}$, приходим к системе линейных алгебраических уравнений с верхней треугольной матрицей $Rx = c$, решение которой совпадает с решением линейной задачи метода наименьших квадратов.

Таким образом, для решения линейной задачи метода наименьших квадратов можно сначала модифицированным методом Грама-Шмидта найти *QR*-разложение матрицы *A*, затем вычислить вектор $c = Q^T b$ и найти $\bar{x}$ как решение системы $Rx = c$.

Для сохранения хороших свойств вычислительной устойчивости вектор $c = (c_1, c_2, \ldots, c_m)^T$ вычисляют иначе. Сначала полагают $r^{(0)} = b$, а затем для $k = 1, \ldots, m$, проводят следующие вычисления

$$c_k = (r^{(k-1)}, q_k), \; r^{(k)} = r^{(k-1)} - c_k q_k .$$

В результате получают вектор *c* и невязку $r = r^{(m)} = b - A\bar{x}$.

Основные вычислительные затраты при решении линейной задачи метода наименьших квадратов производятся на этапе получения *QR*-разложения. В итоге время, необходимое для решения задачи с помощью *QR*-разложения примерно вдвое превышает время, необходимое для решения с использованием нормальной системы.

3.10 Сингулярное разложение матрицы

Пусть A – вещественная матрица размера $n \times m$, где $n \geqslant m$. Тогда существуют такие ортогональные матрицы U и V размера $n \times n$ и $m \times m$ соответственно, что

$$A = U \Sigma V^T .$$

Здесь

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 & ... & 0 \\ 0 & \sigma_2 & 0 & ... & 0 \\ 0 & 0 & \sigma_3 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \sigma_m \\ 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 0 \end{pmatrix}$$

диагональная матрица размера $n \times m$ с элементами $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_3 \geqslant ... \geqslant \sigma_m \geqslant 0$ на главной диагонали.

Разложение называется сингулярным разложением матрицы A или *SVD*-разложением (singular value decomposition). Числа $\sigma_1, ..., \sigma_m$ называются сингулярными числами матрицы A. Эти числа определяются по матрице A однозначно и являются важными ее характеристиками. Например, число ненулевых сингулярных чисел совпадает с рангом матрицы.

Сингулярное разложение используется для решения линейной задачи метода наименьших квадратов. Пусть разложение $A = U \Sigma V^T$ получено. Пусть ранг матрицы A равен *r*, т. е. пусть $\sigma_1 \geqslant \sigma_2 \geqslant \sigma_3 \geqslant ... \geqslant \sigma_r > 0$ и $\sigma_{r+1} = ... = \sigma_m = 0$. Нашей целью является минимизация величины $\|r\|_2 = \|Ax - b\|_2$.

Так как матрица U ортогональна, то норма невязки r не изменится после ее умножения слева на U^T. Таким образом,

$$\|Ax-b\|_2^2=\|U^T(Ax-b)\|_2^2=\|U^T U\Sigma V^T x-U^T b\|_2^2= \\ =\|\Sigma V^T x-U^T b\|_2^2 .$$

Обозначая $y=V^T x$ и $d=U^T b$, мы приходим к эквивалентной задаче минимизации величины

$$\|\Sigma y-d\|_2^2=(\sigma_1 y_1-d_1)^2+(\sigma_2 y_2-d_2)^2+ ... \\ +(\sigma_r y_r-d_r)^2+ d_{r+1}^2+ ...+ d_n^2 .$$

Ясно, что минимум равен $d_{r+1}^2+...+d_n^2$ и достигается он при $y_1=\frac{d_1}{\sigma_1}, ..., y_r=\frac{d_r}{\sigma_r}$. После того как оказался найденным вектор y, осталось положить $x=V y$. Если $r=m$, то решение найдено однозначно. Если же $r<m$, то компоненты $y_{r+1}, ..., y_m$ могут быть выбраны произвольным образом и, следовательно, задача имеет бесконечно много решений. Выбор $y_{r+1}=...=y_m=0$ дает решение x с минимальной нормой (это решение называется псевдорешением системы).

Наличие малых сингулярных чисел σ_i, приводит к тому, что малые погрешности в задании величин d_i приводят к большим погрешностям в вычислении компонент $y_i=\frac{d_i}{\sigma_i}$. В этом случае задача является плохо обусловленной. Правильное использование *SVD* предполагает наличие некоторого допуска $\sigma^*>0$. Все сингулярные числа $\sigma_i \leqslant \sigma^*$ считаются пренебрежимо малыми и заменяются нулями, а соответствующие значения y_i, полагаются равными нулю.

Увеличение значения параметра σ^* приводит к увеличению нормы невязки; однако при этом уменьшается норма решения и оно становится менее чувствительным к входным данным.

Обозначим столбцы матрицы U через $u_1, u_2, , \ldots, u_n$, а столбцы матрицы V через $v_1, v_2, , \ldots, v_m$. Векторы u_i и v_i называются соответственно левыми и правыми сингулярными векторами матрицы A.

Равенство $\|A\bar{x}-b\|_2=\min\limits_{x\in\mathbb{R}^m}\|Ax-b\|_2$ можно записать в виде $AV=U\Sigma$ или $A^TU=V\Sigma^T$. Сравнивая эти матричные равенства по столбцам, видим, что сингулярные векторы обладают следующими свойствами:

$$Av_i=\sigma_i u_i,\ A^Tu_i=\sigma_i v_i,\ 1\leqslant i\leqslant m.$$

Из этих равенств следует, в частности, что

$$A^TAv_i=\sigma_i^2 v_i,\ 1\leqslant i\leqslant m.$$

Таким образом, квадраты сингулярных чисел матрицы A совпадают с собственными значениями матрицы A^TA. Для симметричных матриц сингулярные числа просто равны модулям собственных значений.

Обратим теперь внимание на норму матрицы A. Делая замену $y=Vx$ и учитывая, что умножение вектора на ортогональную матрицу не меняет его евклидову норму, имеем:

$$\|A\|_2=\max_{\|x\|_2=1}\|Ax\|_2=\max_{\|x\|_2=1}\|U^TAx\|_2=\max_{\|x\|_2=1}\|\Sigma V^Tx\|_2=\max_{\|y\|_2=1}\|\Sigma y\|_2=$$
$$=\max_{\|y\|_2=1}\sqrt{\sigma_1^2y_1^2+\ldots+\sigma_m^2y_m^2}=\sigma_1.$$

Отметим также, что отношение $\frac{\sigma_1}{\sigma_m}$ часто используется в качестве числа обусловленности матрицы A. Для квадратной невырожденной матрицы A действительно $cond_2(A)=\frac{\sigma_1}{\sigma_m}$.

Если матрица A хорошо обусловлена, то следует предпочесть метод нормальных уравнений. Однако, если матрица A плохо обусловлена, нормальная система имеет число обусловленности, примерно равное квадрату числа обусловленности матрицы A. Поэтому можно ожидать, что при использовании этого метода будет потеряно вдвое больше значащих цифр по сравнению с методами, основанными на QR разложении или сингулярном разложении матрицы.

В случае, когда матрица A плохо обусловлена, но система её столбцов линейно независима и не близка к линейно зависимой, наиболее подходящим методом является использование QR разложения.

Наиболее медленным, но наиболее надежным методом является использование сингулярного разложения. Особенно важно использование сингулярного разложения в случае, когда система столбцов матрицы A линейно зависима или близка к линейно зависимой.

Таким образом, выбор метода решения зависит от того, что важнее для пользователя: скорость решения задачи или надежность полученных результатов.

3.11 Итерационные методы решения систем линейных алгебраических уравнений

Итерационные методы применяют главным образом для решения задач большой размерности, когда использование прямых методов невозможно из-за ограничений в доступной оперативной

памяти компьютера или из-за необходимости выполнения чрезмерно большого числа арифметических операций. Большие системы уравнений, возникающие в приложениях, как правило, являются разреженными. Методы исключения для решения систем с разреженными матрицами неудобны, например тем, что при их использовании большое число нулевых элементов превращается в ненулевые и матрица теряет свойство разреженности. В противоположность им при использовании итерационных методов в ходе итерационного процесса матрица не меняется. Эффективность итерационных методов по сравнению с прямыми методами тесно связана с возможностью существенного использования разреженности матриц.

Итерационные методы позволяют получать корни системы с заданной точностью путем сходящихся бесконечных процессов. Эффективное применение итерационных методов существенно зависит от удачного выбора начального приближения и быстроты сходимости процесса.

Рассмотрим метод простой итерации. Для того чтобы применить метод простой итерации к решению системы линейных алгебраических уравнений $Ax=b$, с квадратной невырожденной матрицей A, необходимо предварительно преобразовать эту систему к виду

$$x=Bx+c,$$

где

$$B=\begin{pmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mm} \end{pmatrix}, \; c=\begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_m \end{pmatrix}.$$

Самый простой способ приведения системы $Ax=b$ к виду, удобному для итераций, состоит в следующем. Из первого уравнения системы выразим неизвестное x_1, из второго уравнения неизвестное x_2 и т. д. В результате получим систему

$$\begin{aligned}
x_1 &= b_{12}x_2+b_{13}x_3+\ldots+b_{1\,m-1}x_{m-1}+b_{1m}x_m+c_1\\
x_2 &= b_{21}x_1+b_{23}x_3+\ldots+b_{2m-1}x_{m-1}+b_{2m}x_m+c_2\\
x_3 &= b_{31}x_1+b_{32}x_2+\ldots+b_{3\,m-1}x_{m-1}+b_{3m}x_m+c_3\\
\ldots\ \ldots &\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_m &= b_{m1}x_1+b_{m2}x_2+b_{m3}x_3+\ldots+b_{m,\,m-1}x_{m-1}+c_m.
\end{aligned}$$

в которой на главной диагонали матрицы B находятся нулевые элементы. Остальные элементы вычисляются по формулам

$$b_{ij}=-\frac{a_{ij}}{a_{ii}},\ c_i=\frac{b_i}{a_{ii}},\ i,j=1,\ldots,m,\ i\neq j.$$

Для возможности выполнения указанного преобразования необходимо, чтобы диагональные элементы матрицы A были ненулевыми $a_{ii}\neq 0$.

Выберем начальное приближение $x^{(0)}=(x_1^{(0)},\ x_2^{(0)},\ \ldots,x_m^{(0)})$. Подставляя его в правую часть системы $x=Bx+c$ и вычисляя полученное выражение, находим первое приближение

$$x^{(1)}=Bx^{(0)}+c.$$

Подставляя приближение $x^{(1)}$ в правую часть системы получаем $x^{(2)}=Bx^{(1)}+c$. Продолжая этот процесс далее, получаем последовательность $x^{(0)},\ x^{(1)},\ldots,x^{(n)},\ldots$ приближений, вычисляемых по формуле

$$x^{(k+1)}=Bx^{(k)}+c,\ k=0,1,2,\ldots.$$

Если последовательность приближений $x^{(0)}, x^{(1)}, \dots, x^{(n)}, \dots$ имеет предел $x = \lim\limits_{n\to\infty} x^{(n)}$, то этот предел является решением системы.

Процесс итерации хорошо сходится, то есть число приближений, необходимых для получения корней системы с заданной точностью, невелико, если элементы матрицы B малы по абсолютной величине. Иными словами, для успешного применения процесса итерации модули диагональных коэффициентов системы должны быть велики по сравнению с модулями недиагональных коэффициентов этой системы (свободные члены при этом роли не играют).

Метод простой итерации принято называть методом Якоби.

Пусть выполнено условие $\|B\| < 1$, тогда решение x, системы $x = Bx + c$ существует и единственно; при произвольном начальном приближении $x^{(0)}$ метод простой итерации сходится и справедлива оценка погрешности

$$\|x^{(n)} - \bar{x}\| \leqslant \|B\|^n \|x^{(0)} - \bar{x}\| .$$

В практике вычислений в качестве критерия окончания итерационного процесса может быть использовано неравенство $\|x^{(n)} - x^{(n-1)}\| < \epsilon$.

Пример 8. Использовав метод простой итерации в форме Якоби, решить систему с точностью 10^{-3} в норме $\|.\|_\infty$

$$\begin{aligned} 6{,}25\,x_1 - x_2 + 0{,}5\,x_3 &= 7{,}5 \\ -x_1 + 5x_2 + 2{,}12\,x_3 &= -8{,}68 \\ 0{,}5\,x_1 + 2{,}12\,x_2 + 3{,}6\,x_3 &= -0{,}24. \end{aligned}$$

Решение. Вычислив коэффициенты по формулам

$$b_{ij}=-\frac{a_{ij}}{a_{ii}},\ c_i=\frac{b_i}{a_{ii}},\ i,j=1,\ldots,m,\ i\neq j,$$

приведем систему к виду $x=Bx+c$

$$\begin{aligned} x_1 &= 0{,}16\,x_2-0{,}08\,x_3+1{,}2\\ x_2 &= 0{,}2\,x_1-0{,}424\,x_3-1{,}736\\ x_3 &= -0{,}1389\,x_1-0{,}5889\,x_2-0{,}0667. \end{aligned}$$

В последнем уравнении коэффициенты с точностью до погрешности округления. Здесь

$$B=\begin{pmatrix} 0 & 0{,}16 & -0{,}08\\ 0{,}2 & 0 & -0{,}424\\ -0{,}1389 & -0{,}5889 & 0 \end{pmatrix},\ c=\begin{pmatrix} 1{,}2\\ -1{,}736\\ -0{,}0667 \end{pmatrix}.$$

Достаточное условие сходимости метода простой итерации выполнено, так как $\|B\|_\infty=\max\{0{,}24,\ 0{,}624,\ 0{,}7278\}=0{,}7278<1$. Примем за начальное приближение к решению вектор $x^{(0)}=(0,\ 0,\ 0)^T$ и будем вести итерации по формуле $x^{(k+1)}=Bx^{(k)}+c$ до выполнения критерия окончания $\|x^{(n)}-x^{(n-1)}\|<\epsilon$. Значения приближения в таблице 5 приводятся с четырьмя цифрами после десятичной точки.

Таблица 5. Вычисления приближений

n	**0**	**1**	**2**	**3**	**4**	**...**	**14**	**15**
	0,0000	1,2000	0,9276	0,9020	0,8449	...	0,8002	0,8001
	0,0000	-1,7360	-1,4677	-1,8850	-1,8392	...	-1,9995	-1,9998
	0,0000	0,0667	0,7890	0,6688	0,9181	...	0,9995	0,999

При $n=15$ условие $\|x^{(15)}-x^{(14)}\|=0{,}0003<10^{-3}$ выполняется и можно положить $x_1=0{,}8000\pm0{,}001$, $x_2=-2{,}000\pm0{,}001$, $x_3=1{,}000\pm0{,}001$. Точные значения решения нам известны $x_1=0{,}8$, $x_2=-2$, $x_3=1$.

Пусть $\lambda_1, \lambda_2, \ldots, \lambda_m$ собственные значения матрицы B. Спектральным радиусом матрицы B называется величина $\rho(B)=\max\limits_{1\leqslant i\leqslant m}|\lambda_i|$.

Метод простой итерации сходится при любом начальном приближении $x^{(0)}$ тогда и только тогда, когда $\rho(B)<1$, т. е. когда все собственные значения матрицы B по модулю меньше единицы.

Для симметричной матрицы B ее спектральный радиус $\rho(B)$ совпадает с $\|B\|_2$. Поэтому для сходимости метода простой итерации с симметричной матрицей B условие $\|B\|_2<1$ является необходимым и достаточным.

Модификацией метода Якоби является метод Зейделя. Пусть система $Ax=b$ приведена к виду

$$\begin{aligned}
x_1 &= b_{12}x_2+b_{13}x_3+\ldots+b_{1\,m-1}x_{m-1}+b_{1m}x_m+c_1\\
x_2 &= b_{21}x_1+b_{23}x_3+\ldots+b_{2m-1}x_{m-1}+b_{2m}x_m+c_2\\
x_3 &= b_{31}x_1+b_{32}x_2+\ldots+b_{3\,m-1}x_{m-1}+b_{3m}x_m+c_3\\
\ldots\ &\ldots\ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
x_m &= b_{m1}x_1+b_{m2}x_2+b_{m3}x_3+\ldots+b_{m,\,m-1}x_{m-1}+c_m.
\end{aligned}$$

с коэффициентами, вычисленными по формулам

$$b_{ij}=-\frac{a_{ij}}{a_{ii}},\ c_i=\frac{b_i}{a_{ii}},\ i,j=1,\ldots,m,\ i\neq j.$$

Основная идея модификации состоит в том, что при вычислении очередного (k+1)-го приближения к неизвестному x_i при $i>1$,

используют уже найденные (k+1)-е приближения к неизвестным $x_1, \ldots, x_{i-1}$, а не k-е приближения, как в методе Якоби. На (k+1)-й итерации компоненты приближения x^{k+1} вычисляются по формулам

$$\begin{aligned}
x_1^{(k+1)} &= b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \ldots + b_{1\,m-1}x_{m-1}^{(k)} + b_{1\,m}x_m^{(k)} + c_1 \\
x_2^{(k+1)} &= b_{21}x_1^{(k+1)} + b_{23}x_3^{(k)} + \ldots + b_{2\,m-1}x_{m-1}^{(k)} + b_{2\,m}x_m^{(k)} + c_2 \\
x_3^{(k+1)} &= b_{31}x_1^{(k+1)} + b_{32}x_2^{(k+1)} + \ldots + b_{3\,m-1}x_{m-1}^{(k)} + b_{3\,m}x_m^{(k)} + c_3 \\
\ldots\ \ldots\ & \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
x_m^{(k+1)} &= b_{m1}x_1^{(k+1)} + b_{m2}x_2^{(k+1)} + \ldots + b_{m,\,m-1}x_{m-1}^{(k+1)} + c_m
\end{aligned}$$

Введем нижнюю и верхнюю треугольные матрицы

$$B_1 = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \\ b_{21} & 0 & 0 & \ldots & 0 \\ b_{31} & b_{32} & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ b_{m1} & b_{m2} & b_{m3} & \ldots & 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 & b_{12} & b_{13} & \ldots & b_{1m} \\ 0 & 0 & b_{23} & \ldots & b_{2m} \\ 0 & 0 & 0 & \ldots & b_{3m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}.$$

Тогда расчетные формулы метода примут вид: $x^{(k+1)} = B_1 x^{(k+1)} + B_2 x^{(k)} + c$.

Заметим, что $B = B_1 + B_2$ и поэтому решение $\bar{x}$ исходной системы удовлетворяет равенству $\bar{x} = B_1\bar{x} + B_2\bar{x} + c$. Метод Зейделя иногда называют также методом Гаусса-Зейделя, процессом Либмана, методом последовательных замещений.

Пусть $\|B\| < 1$, где $\|B\|$ одна из норм $\|B\|_\infty$, $\|B\|_1$. Тогда при любом выборе начального приближения $x^{(0)}$ метод Зейделя сходится со скоростью геометрической прогрессии, знаменатель которой $q \leqslant \|B\|$.

Пусть A – симметричная положительно определенная матрица. Тогда при любом выборе начального приближения $x^{(0)}$ метод Зейделя сходится со скоростью геометрической прогрессии.

Если требуется найти решение с точностью ϵ, то итерации метода Зейделя следует вести до выполнения неравенства $\|x^{(n)} - x^{(n-1)}\| < \epsilon$.

Пример 9. Использовав метод Зейделя, решить систему с точностью 10^{-3} в норме $\|.\|_\infty$

$$\begin{aligned} 6{,}25\,x_1 - x_2 + 0{,}5\,x_3 &= 7{,}5 \\ -x_1 + 5\,x_2 + 2{,}12\,x_3 &= -8{,}68 \\ 0{,}5\,x_1 + 2{,}12\,x_2 + 3{,}6\,x_3 &= -0{,}24 \end{aligned}$$

Решение. Вычислив коэффициенты по формулам

$$b_{ij} = -\frac{a_{ij}}{a_{ii}},\ c_i = \frac{b_i}{a_{ii}},\ i, j = 1, \ldots, m,\ i \neq j,$$

приведем систему к виду $x = Bx + c$

$$\begin{aligned} x_1 &= 0{,}16\,x_2 - 0{,}08\,x_3 + 1{,}2 \\ x_2 &= 0{,}2\,x_1 - 0{,}424\,x_3 - 1{,}736 \\ x_3 &= -0{,}1389\,x_1 - 0{,}5889\,x_2 - 0{,}0667 \end{aligned}$$

В последнем уравнении коэффициенты с точностью до погрешности округления. Здесь

$$B = \begin{pmatrix} 0 & 0{,}16 & -0{,}08 \\ 0{,}2 & 0 & -0{,}424 \\ -0{,}1389 & -0{,}5889 & 0 \end{pmatrix},\ c = \begin{pmatrix} 1{,}2 \\ -1{,}736 \\ -0{,}0667 \end{pmatrix}.$$

Достаточное условие сходимости $\|B\|_\infty = \max\{0{,}24,\ 0{,}624,\ 0{,}7278\} = 0{,}7278 < 1$ выполнено, поэтому метод Зейделя сходится. Примем за начальное приближение к решению вектор $x^{(0)} = (0, 0, 0)^T$ и будем вести итерации по

формулам $\begin{aligned} x_1^{(k+1)} &= 0{,}16\,x_2^k - 0{,}08\,x_3^k + \ldots + 1{,}2 \\ x_2^{(k+1)} &= 0{,}2\,x_1^{(k+1)} - 0{,}424\,x_3^k - 1{,}736 \\ x_3^{(k+1)} &= -0{,}1389\,x_1^{(k+1)} - 0{,}5889\,x_2^{(k+1)} - 0{,}0667 \end{aligned}$. Здесь

$$B_2 = \begin{pmatrix} 0 & 0{,}16 & -0{,}08 \\ 0 & 0 & -0{,}424 \\ 0 & 0 & 0 \end{pmatrix} \text{ и } \|B_2\|_\infty = \max\{0{,}24,\ 0{,}424,\ 0\} = 0{,}424 < 1 .$$

Будем вести итерации до выполнения критерия окончания $\|x^{(n)} - x^{(n-1)}\| < \epsilon$. Значения приближения в таблице 6 приводятся с четырьмя цифрами после десятичной точки.

Таблица 6. Значения приближений

n	**0**	**1**	**2**	**3**	**...**	**7**	**8**
	0,0000	1,2000	0,9088	0,8367	...	0,8004	0,8001
	0,0000	-1,4960	-1,8288	-1,9435	...	-1,9993	-1,9998
	0,0000	0,64766	0,8841	0,9616	...	0,9995	0,9998

При $n = 8$ условие $\|x^{(8)} - x^{(7)}\| = 0{,}0005 < 10^{-3}$ выполняется и можно положить $x_1 = 0{,}8000 \pm 0{,}001$, $x_2 = -2{,}000 \pm 0{,}001$, $x_3 = 1{,}000 \pm 0{,}001$.

Точные значения решения нам известны

$$x_1 = 0{,}8,\ x_2 = -2,\ x_3 = 1 .$$

Итерационный метод называют одно шаговым или двухслойным если для вычисления очередного приближения $x^{(k+1)}$ к решению $\bar{x}$ системы используется только одно предыдущее приближение $x^{(k)}$.

Многие двухслойные итерационные методы могут быть записаны в каноническом виде

$$B\frac{x^{(k+1)}+x^{(k)}}{\tau_{k+1}}+Ax^{(k)}=b,\ k=0,1,\ldots$$

Здесь B - некоторая невырожденная матрица, $\tau_{k+1}>0$ – итерационные параметры. Матрица B может, вообще говоря, зависеть от k, то есть меняться от итерации к итерации. В случае, когда параметры $\tau_{k+1}=\tau$ (и матрица B) не зависят от номера итерации, метод называют стационарным.

Если $B=E$, итерационный метод принимает вид

$$\frac{x^{(k+1)}+x^{(k)}}{\tau_{k+1}}+Ax^{(k)}=b,\ k=0,1,\ldots$$

и называется явным, поскольку в нем очередное приближение явным образом выражается через предыдущее

$$x^{(k+1)}=x^{(k)}-\tau_{k+1}\left(Ax^{(k)}-b\right),\ k=0,1,\ldots.$$

В общем случае, при $B\neq E$, метод является неявным, так как на каждой итерации требуется решать систему уравнений

$$Bx^{(k+1)}=Bx^{(k)}-\tau_{k+1}\left(Ax^{(k)}-b\right),\ k=0,1,\ldots.$$

Конечно, матрица B должна быть такой, чтобы каждая из этих систем решалась значительно быстрее, чем исходная система $Ax=b$. Например, матрица B может быть диагональной, трехдиагональной, треугольной или разреженной.

При численной реализации неявного метода обычно сначала вычисляется невязка $r^{(k)}=b-Ax^{(k)}$, затем относительно вектора $w^{(k)}=\frac{x^{(k+1)}-x^{k}}{\tau_{K+1}}$, пропорционального поправке к решению, решается система уравнений $Bw^{(k)}=r^{(k)}$, после чего находится очередное приближение $x^{(k+1)}=x^{(k)}-\tau_{k+1}w^{(k)}$.

3.12 Упражнения

1. Следующие системы решить методом Гаусса с выбором главного элемента и обычным методом Гаусса, проведя все вычисления с пятью значащими цифрами. Сравнить полученные значения с указанными точными значениями.

a) $A=\begin{pmatrix}0{,}15 & 2{,}11 & 30{,}75\\ 0{,}64 & 1{,}21 & 2{,}05\\ 3{,}21 & 1{,}53 & 1{,}04\end{pmatrix}$, $b=\begin{pmatrix}-26{,}38\\ 1{,}01\\ 5{,}23\end{pmatrix}$, $x=\begin{pmatrix}1\\ 2\\ -1\end{pmatrix}$;

b) $A=\begin{pmatrix}1{,}15 & 0{,}42 & 100{,}71\\ 1{,}19 & 0{,}55 & 0{,}32\\ 1{,}00 & 0{,}35 & 3{,}00\end{pmatrix}$, $b=\begin{pmatrix}-198{,}70\\ 2{,}29\\ -0{,}65\end{pmatrix}$, $x=\begin{pmatrix}2\\ 1\\ -2\end{pmatrix}$.

2.Решить системы, пользуясь методом Холецкого.

a) $A=\begin{pmatrix}2{,}5 & -3{,}0 & 4{,}6\\ -3{,}5 & 2{,}6 & 1{,}5\\ -6{,}5 & -3{,}5 & 7{,}3\end{pmatrix}$, $b=\begin{pmatrix}-1{,}05\\ -14{,}46\\ -17{,}735\end{pmatrix}$;

b) $A=\begin{pmatrix}2 & -1 & 4 & -3 & 1\\ -1 & 1 & 2 & 1 & 3\\ 4 & 2 & 3 & 3 & -1\\ -3 & 1 & 3 & 2 & 4\\ 1 & 3 & -1 & 4 & 4\end{pmatrix}$, $b=\begin{pmatrix}11\\ 14\\ 4\\ 16\\ 18\end{pmatrix}$.

3.Методом квадратных корней решить системы уравнений. Вычисления вести с пятью знаками после запятой.

a) $A=\begin{pmatrix}3{,}1 & 1{,}5 & 1{,}0\\ 1{,}5 & 2{,}5 & 0{,}5\\ 1{,}0 & 0{,}5 & 4{,}2\end{pmatrix}$, $b=\begin{pmatrix}10{,}83\\ 9{,}20\\ 17{,}10\end{pmatrix}$;

b) $A=\begin{pmatrix}3,2 & 1 & 1\\ 1 & 3,7 & 1\\ 1 & 1 & 4,2\end{pmatrix}$, $b=\begin{pmatrix}4\\ 4,5\\ 5\end{pmatrix}$.

4.Решить системы методом простой итерации. Продолжать итерации до тех пор, пока разница между последовательными приближениями неизвестных не станет меньше 10^{-3}. Сравнить ответ с данными точными значениями неизвестных.

a) $A=\begin{pmatrix}0,15 & 2,11 & 30,75\\ 0,64 & 1,21 & 2,05\\ 3,21 & 1,53 & 1,04\end{pmatrix}$, $b=\begin{pmatrix}-26,38\\ 1,01\\ 5,23\end{pmatrix}$, $x=\begin{pmatrix}1\\ 2\\ -1\end{pmatrix}$;

b) $A=\begin{pmatrix}10,9 & 1,2 & 2,1 & 0,9\\ 1,2 & 11,2 & 1,5 & 2,5\\ 2,1 & 1,5 & 9,8 & 1,3\\ 0,9 & 2,5 & 1,3 & 12,1\end{pmatrix}$, $b=\begin{pmatrix}-7,0\\ 5,3\\ 10,3\\ 24,6\end{pmatrix}$, $x=\begin{pmatrix}-1\\ 0\\ 1\\ 2\end{pmatrix}$.

5.Системы решить методом простой итерации и методом Зейделя. Сравнить скорости сходимости итераций. Полученные значения сравнить с указанными точными значениями неизвестных

a) $A=\begin{pmatrix}6,1 & 2,2 & 1,2\\ 2,2 & 5,5 & -1,5\\ 1,2 & -1,5 & 7,2\end{pmatrix}$, $b=\begin{pmatrix}16,55\\ 10,55\\ 16,80\end{pmatrix}$, $x=\begin{pmatrix}1,5\\ 2,0\\ 2,5\end{pmatrix}$;

b) $A=\begin{pmatrix}3,82 & 1,02 & 0,75 & 0,81\\ 1,05 & 4,53 & 0,98 & 1,53\\ 0,73 & 0,85 & 4,71 & 0,81\\ 0,88 & 0,81 & 1,28 & 3,50\end{pmatrix}$, $b=\begin{pmatrix}15,655\\ 22,705\\ 23,480\\ 16,110\end{pmatrix}$, $x=\begin{pmatrix}2,5\\ 3,0\\ 3,5\\ 2,0\end{pmatrix}$.

3.13 Лабораторная работа «Решение СЛАУ»

Везде далее рассматриваем решение системы линейных уравнений вида:

$$\begin{cases} a_{11}x_1+a_{12}x_2+...+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+...+a_{2n}x_n=b_2 \\ ... \\ a_{n1}x_1+a_{n2}x_2+...+a_{nn}x_n=b_n \end{cases}$$ или в матричном виде: $A \cdot X = B$,

где на матрицу A будем накладывать определенные условия. По умолчанию будем считать, что определитель матрицы A не равен нулю.

Из курса линейной алгебры вам известны методы точного нахождения решения системы: метод Крамера, метод Гаусса, метод обратной матрицы.

В большинстве случаев решение линейной системы уравнений «руками» представляет определенные вычислительные сложности и становится труднореализуемо при большом количестве неизвестных. В данном курсе вы познакомитесь с численными методами решения линейных систем. Методы решения систем можно условно разделить на итерационные методы и неитерационные. Познакомимся сначала с неитерационными методами.

Решение треугольных систем линейных уравнений.

Наиболее просто исходная система линейных уравнений решается, когда матрица A приведена к треугольному виду с единицами по главной диагонали (к унитреугольному виду). Напомню, что матрицу A всегда можно привести к подобному виду, если определитель матрицы A не равен 0. если система уже приведена к виду:

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ a_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix},$$

то её можно решить прямой подстановкой:

$$\begin{cases} x_1 = b_1 \\ x_2 = b_2 - a_{21} x_1 \\ \dots \\ x_n = b_n - a_{n1} x_1 - a_{n2} x_2 - \dots - a_{nn} x_n. \end{cases}$$

Реализация метода приведена на рисунке 7:

```
A = np.array([[1, 0, 0],
              [3, 1, 0],
              [-4, 5, 1]])#Левая часть системы
B = np.array([2, 4, 3])#Правая часть системы
x = np.zeros((3,1)) # вектор решений

x[0] = B[0]

for i in range(1, 3):
    x[i] = B[i] - np.dot(A[i, :i], x[ :i])

print(x)
```

```
[[ 2.]
 [-2.]
 [21.]]
```

Рисунок 7. Решение унитреугольной системы

В цикле умножаем элементы *i*-той строки матрицы *A* до главной диагонали на часть вектора *x*. Также возможно решение без введения переменной *x*, хранящей решение системы.

Верхнетреугольная система решается методом обратной подстановки:

$$\begin{pmatrix} 1 & a_{12} & \dots & a_{1n} \\ 0 & 1 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \Rightarrow \begin{cases} x_n = b_n \\ x_{n-1} = b_{n-1} - a_{n1} x_n \\ \dots \\ x_1 = b_1 - a_{12} x_2 - a_{13} x_3 - \dots - a_{1n} x_n. \end{cases}$$

Аналогично можно решить треугольные системы с диагональными элементами не равными нулю.

Применение *LU* разложения к решению систем линейных уравнений.

Если известно *LU* разложение матрицы, то исходная система может быть записана как: *LUx=b*. Эта система может быть решена в два шага. На первом шаге решается система *Ly=b*. Поскольку *L* – нижняя унитреугольная матрица, эта система решается непосредственно прямой подстановкой. На втором шаге решается система *Ux=y*. Поскольку *U* – верхняя треугольная матрица, эта система решается непосредственно обратной подстановкой. Реализовать этот метод предлагается самостоятельно. Рекомендуется использовать функции np.tril и np.triu. для создания треугольных матриц. Np.tril(m, k=0) – возвращает копию массива m, с элементами выше *k*-той диагонали равными нулю. По умолчанию *k*=0 (главная диагональ), *k*<0 диагональ ниже главной, *k*>0 диагональ выше главной диагонали. Np.triu(m, k=0) – возвращает копию массива *m*, с элементами выше *k*-той диагонали равными нулю.

Применение *QR* разложения к решению систем линейных уравнений.

Пусть матрица *A* имеет размеры $n \times n$. Представим матрицу *A* в виде *A=QR*, где матрица *Q* – квадратная матрица $n \times n$ с ортонормированными столбцами, матрица *R* – квадратная $n \times n$ верхнетреугольная матрица. Ортонормированность столбцов мат-

рицы Q равносильно выполнению условия $Q^TQ=E$. Допустим, что QR разложение матрицы A известно.

Преобразуем систему:

$AX=B\Rightarrow QRX=B\Rightarrow Q^TQRX=Q^{TB}\Rightarrow RX=Q^T B$. Свели задачу к решению верхнетреугольной системы ,которая решается обратной подстановкой.

Метод простых итераций (Якоби).

Напомним, что исходная система имеет вид:

$$\begin{cases} a_{11}x_1+a_{12}x_2+...+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+...+a_{2n}x_n=b_2 \\ ... \\ a_{n1}x_1+a_{n2}x_2+...+a_{nn}x_n=b_n. \end{cases}$$

Преобразуем систему, выразив из первого уравнения x_1, из второго x_2 и так далее:

$$\begin{cases} a_{11}x_1=b_1-a_{12}x_2-...-a_{1n}x_n \\ a_{22}x_2=b_2-a_{21}x_1-...-a_{2n}x_n \\ ... \\ a_{nn}x_n=b_n-a_{n1}x_1-a_{n2}x_2-...+a_{nn-1}x_{n-1} \end{cases} \Rightarrow$$

$$\begin{cases} x_1=1/a_{11}(b_1-a_{12}x_2-...-a_{1n}x_n) \\ x_2=1/a_{22}(b_2-a_{21}x_1-...-a_{2n}x_n) \\ ... \\ x_n=1/a_{nn}(b_n-a_{n1}x_1-a_{n2}x_2-...+a_{nn-1}x_{n-1}). \end{cases}$$

Итерационный процесс:

$$\begin{cases} x_1^{k+1}=1/a_{11}(b_1-a_{12}x_2^k-...-a_{1n}x_n^k) \\ x_2^{k+1}=1/a_{22}(b_2-a_{21}x_1^k-...-a_{2n}x_n^k) \\ ... \\ x_n^{k+1}=1/a_{nn}(b_n-a_{n1}x_1^k-a_{n2}x_2^k-...+a_{nn-1}x_{n-1}^k). \end{cases}$$

В матричном виде: $X^{k+1}=BX^k+C$, где B квадратная матрица с нулями по главной диагонали, на остальных местах элементы вида $b_{ij}=-\frac{a_{ij}}{a_{ii}}$, C – вектор-столбец с элементами $c_i=\frac{b_i}{a_{ii}}$. Заметим, что такое преобразование возможно только в том случае, если диагональные элементы исходной матрицы A не равны нулю. Такой метод преобразования матрицы A называют методом Якоби. Возможны другие методы преобразования матрицы A ,но их рассмотрение выходит за рамки данного курса. Выберем начальное приближение $X^0=(x_1^0, x_2^0, ..., x_n^0)^T$. Запустим итерационный процесс: $X^1=B\cdot X^0+C$. Далее подставим найденное приближение: $X^2=B\cdot X^1+C$ и так далее. Получим последовательность приближенных решений: $X^0, X^1, X^2 ...$. Достаточным условием сходимости последовательности приближенных решений к точному является условие диагонального преобладания матрицы А: $\sum_{j,\, j\neq i}|a_{ij}|<|a_{ii}|\,(для\ любой\ строки\ i)$ или вытекающие условие для матрицы B: $\|B\|_1<1$ $или$ $\sum_j|b_{ij}|<1\,(для\ любой\ строки\ i)$, также можно показать, что для сходимости метода Якоби достаточно, чтобы любая норма матрицы B была меньше 1. Заметим, что если исходная матрица A не удовлетворяет условию диагонального преобладания, достаточно, применив эквивалентные преобразования системы, привести ее к нужному виду, после чего найти матрицу B, которая будет удовлетворять условию $\|B\|<1$. Далее запускаем итерационный процесс $X^{k+1}=B\cdot X^k+C$, где в качестве начального приближения можно выбрать любой вектор $X^{0.}$. В учебных заданиях можно выбирать нулевой вектор. Условием остановки итерационного процесса может служить условие: $\|X^{k+1}-X^k\|_\infty<\epsilon$, где ϵ заданная точность.

Пример. Решим систему $\begin{cases} 100x_1+30x_2-70x_3=60 \\ 15x_1-50x_2-5x_3=-40 \\ 6x_1+2x_2+20x_3=28 \end{cases}$ с точностью до $\epsilon=0.01$. Заметим, что матрица $A=\begin{pmatrix} 100 & 30 & -70 \\ 15 & -50 & -5 \\ 6 & 2 & 20 \end{pmatrix}$ не удовлетворяет условию диагонального преобладания, но в данной системе достаточно прибавить к первому уравнению второе:

$$\begin{cases} 115x_1-20x_2-75x_3=20 \\ 15x_1-50x_2-5x_3=-40 \\ 6x_1+2x_2+20x_3=28 \end{cases} \Rightarrow \begin{cases} x_1=20/115+20/115\,x_2+75/115\,x_3 \\ x_2=40/50+15/50x_1-5/50x_3 \\ x_3=28/20-6/20x_1-2/20x_2. \end{cases}$$

Получили систему, удовлетворяющую достаточному условию сходимости метода Якоби. Результат выполнения итераций представлен на рисунке:

k	x_1	x_2	x_3	norma(x_k- x_k-1)
0	0.0	0.0	0.0	
1	0.17391304347826086	0.8	1.4	1.4
2	1.2260869565217392	0.7121739130434783	1.2678260869565217	1.0521739130434784
3	1.1246124763705103	1.0410434782608697	0.9609565217391303	0.3288695652173914
4	0.9816748582230623	1.0412880907372402	0.9585119092627599	0.142937618147448
5	0.9801230870387113	0.9986512665406427	1.0013687334593573	0.04285682419659742
6	1.0006580899153448	0.9939000527656776	1.0060979472343223	0.020535002876633457
7	1.0029160617207629	0.9995876322511712	1.0004125677488287	0.005687579485493588
8	1.0001973497929182	1.000833561741346	0.9991664182586539	0.0027187119278446747

Рисунок 8. Пример работы методы Якоби

В левом столбце указан номер итерации. В последней строке набор координат (x_1, x_2, x_3) является приближенным решением исходной системы, найденным с указанной точностью. В последнем столбце выводится норма разности текущей итерации и предыдущей (максимум из модулей координат). Вывод таблицы оформлен в Python с помощью модуля PrettyTable. В лабораторной

работе допускаются и другие варианты оформления вывода результатов.

Метод Зейделя.

Метод Зейделя является модификацией метода Якоби. В отличии от метода Якоби, в методе Зейделя при вычислении x_i^{k+1} используются значения x_1^{k+1}, x_2^{k+1}, x_{i-1}^{k+1}, уже найденные на (k+1)-ой итерации, т. е. (k+1)-е приближение строится следующим образом:

$$\begin{cases} x_1^{k+1} = 1/a_{11}(b_1 - a_{12}x_2^k - \ldots - a_{1n}x_n^k) \\ x_2^{k+1} = 1/a_{22}(b_2 - a_{21}x_1^{k+1} - a_{23}x_3^k \ldots - a_{2n}x_n^k) \\ \ldots \\ x_n^{k+1} = 1/a_{nn}(b_n - a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \ldots + a_{nn-1}x_{n-1}^{k+1}) \end{cases}$$

В матричном виде:

$$\begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ \vdots \\ x_n^{k+1} \end{pmatrix} = \begin{pmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \ldots & 0 \\ -a_{21}/a_{22} & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ -a_{n1}/a_{nn} & -a_{n2}/a_{nn} & \ldots & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1^{k+1} \\ x_2^{k+1} \\ \vdots \\ x_n^{k+1} \end{pmatrix} +$$

$$+ \begin{pmatrix} 0 & -a_{12}/a_{11} & \ldots & -a_{1n}/a_{11} \\ 0 & 0 & \ldots & -a_{2n}/a_{22} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1^k \\ x_2^k \\ \vdots \\ x_n^k \end{pmatrix}.$$

Проще говоря: $X^k = C + B_1 X^{k+1} + B_2 X^k$, где $B = B_1 + B_2$, B_1 – нижнетреугольная матрица, B_2 – верхнетреугольная матрица. Для сходимости метода Зейделя достаточно, чтобы $max|b_{ij}| < 1$, то есть чтобы максимальный по модулю элемент матрицы B был меньше 1. Условием остановки итерационного процесса может служить

условие: $\|X^{k+1}-X^k\|_\infty<\epsilon$, где ϵ заданная точность. Метод Зейделя, как правило, сходится быстрее, чем метод Якоби.

Пример.

Решим пример из предыдущего пункта методом Зейделя

$$\begin{cases} x_1^{k+1}=20/115+20/115\,x_2^k+75/115\,x_3^k \\ x_2^{k+1}=40/50+15/50\,x_1^{k+1}-5/50\,x_3^k \\ x_3^{k+1}=28/20-6/20\,x_1^{k+1}-2/20\,x_2^{k+1} \end{cases}$$

$$X^{k+1}=\begin{pmatrix} 20/115 \\ 40/50 \\ 28/20 \end{pmatrix}+\begin{pmatrix} 0 & 0 & 0 \\ 15/50 & 0 & 0 \\ -6/20 & -2/20 & 0 \end{pmatrix}\cdot X^{k+1}+$$

$$+\begin{pmatrix} 0 & 20/115 & 75/115 \\ 0 & 0 & -5/50 \\ 0 & 0 & 0 \end{pmatrix}\cdot X^k.$$

Результат выполнения итераций представлен на рисунке:

k	x_1	x_2	x_3	norma(x_k- x_k-1)
0	0.0	0.0	0.0	
1	0.9996013269932693	1.00014256311201	0.9998574388879898	1.00014256311201
2	0.9999318189464299	0.9998946542091818	1.0001053455908182	0.0003304919531605943

Рисунок 9. Метод Зейделя

Для этого примера метод Зейделя сошелся гораздо быстрее метода Якоби.

Решение переопределенных систем методом псевдообратной матрицы.

В этом пункте предположим, что количество уравнений больше количества неизвестных, линейная система имеет вид: $A\cdot X=B$, где A – матрица размера $m\times n$, X – столбец $n\times 1$, B имеет размер $m\times 1$. Обратной матрицы для матрицы A не существует, однако, если допустить, что столбцы A линейно независимы, то тогда существует матрица к матрице $A^T\cdot A$. Умножим левую и правую часть уравнения на A^T:

$$A^T \cdot A \cdot X = A^T \cdot B \Rightarrow (A^T \cdot A)^{-1} \cdot A^T \cdot A \cdot X = (A^T \cdot A)^{-1} \cdot A^T \cdot B .$$

Откуда следует, что $X = (A^T \cdot A)^{-1} \cdot A^T \cdot B$ – псевдорешение исходной системы. Матрица $(A^T \cdot A)^{-1} \cdot A^T$ называется псевдообратной матрицей к матрице A. Псевдорешение – это приближенное решение исходной системы, которое дает минимальную евклидову норму невязки $\|AX - B\|_2$.

Вопросы к лабораторной работе.

1. Определение линейной системы уравнений.
2. Методы решения систем: метод Крамера, метод Гаусса, метод обратной матрицы.
3. Верхнетреугольная, нижнетреугольная, унитреугольная матрицы.
4. Определение *LU* разложения, определение QR разложения матрицы.
5. Идея решения системы линейных уравнений с помощью *LU* разложения, с помощью *QR* разложения.
6. Итерационные методы решения линейных систем. Достаточное условие сходимости метода Якоби, метода Зейделя.
7. Поиск решения переопределенной системы линейных уравнений с помощью обратной матрицы.
8. Нормы векторов и матриц, используемые при решении линейных систем.

Задание к лабораторной работе.

Вариант 1.

1. Создать произвольную верхнетреугольную матрицу A 5 порядка (не унитреугольную), вектор B произвольный. Решить си-

стему AX = B методом обратной подстановки. Проверить решение методом np.solve.

2. Решить систему, используя LU разложение

$$\begin{cases} 4.4x_1 - 2.5x_2 + 19.2x_3 - 10.8x_4 = 4.3 \\ 5.5x_1 - 9.3x_2 - 14.2x_3 + 13.2x_4 = 6.8 \\ 7.1x_1 - 11.5x_2 + 5.3x_3 - 6.7x_4 = -1.8 \\ 14.2x_1 + 23.4x_2 - 8.8x_3 + 5.3x_4 = 7.2 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Хаусхолдера. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas). Проверить полученное решение.

$$\begin{cases} 2.7x_1 + 3.3x_2 + 1.3x_3 = 2.1 \\ 3.5x_1 - 1.7x_2 + 2.8x_3 = 1.7 \\ 4.1x_1 + 5.8x_2 - 1.7x_3 = 0.8 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 4.4x_1 - 2.5x_2 + 19.2x_3 - 10.8x_4 = 4.3 \\ 5.5x_1 - 9.3x_2 - 14.2x_3 + 13.2x_4 = 6.8 \\ 7.1x_1 - 11.5x_2 + 5.3x_3 - 6.7x_4 = -1.8 \\ 14.2x_1 + 23.4x_2 - 8.8x_3 + 5.3x_4 = 7.2 \\ 8.2x_1 - 3.2x_2 + 14.2x_3 + 14.8x_4 = -8.4 \end{cases}$$

Вариант 2.

1. Создать произвольную нижнетреугольную матрицу A 5 порядка (не унитреугольную), вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 8.2\,x_1 - 3.2\,x_2 + 14.2\,x_3 + 14.8\,x_4 = -8.4 \\ 5.6\,x_1 - 12\,x_2 + 15\,x_3 - 6.4\,x_4 = 4.5 \\ 5.7\,x_1 + 3.6\,x_2 - 12.4\,x_3 - 2.3\,x_4 = 3.3 \\ 6.8\,x_1 + 13.2\,x_2 - 6.3\,x_3 - 8.7\,x_4 = 14.3 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Грама-Шмидта. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas). Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

$$\begin{cases} 2.7\,x_1 + 3.3\,x_2 + 1.3\,x_3 = 2.1 \\ 3.5\,x_1 - 1.7\,x_2 + 2.8\,x_3 = 1.7 \\ 4.1\,x_1 + 5.8\,x_2 - 1.7\,x_3 = 0.8 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 3.1x_1+2.8x_2+1.9x_3=0.2 \\ 1.9x_1+3.1x_2+2.1x_3=2.1 \\ 7.5x_1+3.8x_2+4.8x_3=5.6 \\ 3.01x_1-0.33x_2+0.11x_3=0.13 \end{cases}$$

Вариант 3.

1. Создать произвольную верхнюю унитреугольную матрицу A 5 порядка, вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 5.7x_1-7.8x_2-5.6x_3-8.3x_4=2.7 \\ 6.6x_1+13.1x_2-6.3x_3+4.3x_4=-5.5 \\ 14.7x_1-2.8x_2+5.6x_3-12.1x_4=8.6 \\ 8.5x_1+12.7x_2-23.7x_3+5.7x_4=14.7 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Гивенса. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом Зейделя с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 0.21x_1-0.18x_2+0.75x_3=0.11 \\ 0.13x_1+0.75x_2-0.11x_3=2 \\ 3.01x_1-0.33x_2+0.11x_3=0.13 \end{cases}$$

Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

5. Найти псевдорешение системы

$$\begin{cases} 5.7x_1 - 7.8x_2 - 5.6x_3 = 2.7 \\ 6.6x_1 + 13.1x_2 - 6.3x_3 = -5.5 \\ 14.7x_1 - 2.8x_2 + 5.6x_3 = 8.6 \\ 8.5x_1 + 12.7x_2 - 23.7x_3 = 14.7 \end{cases}$$

Вариант 4.

1. Создать произвольную нижнюю унитреугольную матрицу A 7 порядка, вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 3.8x_1 + 14.2x_2 + 6.3x_3 - 15.5x_4 = 2.8 \\ 8.3x_1 - 6.6x_2 + 5.8x_3 + 12.2x_4 = -4.7 \\ 6.4x_1 - 8.5x_2 - 4.3x_3 + 8.8x_4 = 7.7 \\ 17.1x_1 - 8.3x_2 + 14.4x_3 - 7.2x_4 = 13.5 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Хаусхолдера. Проверить полученное разложение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas). Проверить полученное решение. $\begin{cases} 3.3x_1 + 2.1x_2 + 2.8x_3 = 0.8 \\ 4.1x_1 + 3.7x_2 + 4.8x_3 = 5.7 \\ 2.7x_1 + 1.8x_2 + 1.1x_3 = 3.2 \end{cases}$

5. Найти псевдорешение системы

$$\begin{cases} 3.3x_1+2.1x_2+2.8x_3=0.8 \\ 4.1x_1+3.7x_2+4.8x_3=5.7 \\ 2.7x_1+1.8x_2+1.1x_3=3.2 \\ 3.15x_1-1.72x_2-1.23x_3=2.15 \end{cases}$$

Вариант 5.

1. Создать произвольную верхнетреугольную матрицу A 4 порядка (не унитреугольную), вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 15.7x_1+6.6x_2-5.7x_3+11.5x_4=-2.4 \\ 8.8x_1-6.7x_2+5.5x_3-4.5x_4=5.6 \\ 6.3x_1-5.7x_2-23.4x_3+6.6x_4=7.7 \\ 14.3x_1+8.7x_2-15.7x_3-5.8x_4=23.4 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Грама-Шмидта. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом Зейделя с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 3.15x_1-1.72x_2-1.23x_3=2.15 \\ 0.72x_1+0.67x_2+1.18x_3=1.43 \\ 2.57x_1-1.34x_2-0.68x_3=1.03 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 15.7x_1+6.6x_2-5.7x_3=-2.4 \\ 8.8x_1-6.7x_2+5.5x_3=5.6 \\ 6.3x_1-5.7x_2-23.4x_3=7.7 \\ 14.3x_1+8.7x_2-15.7x_3=23.4 \end{cases}$$

Вариант 6.

1. Создать произвольную верхнетреугольную матрицу A 5 порядка (не унитреугольную), вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 4.3x_1-12.1x_2+23.2x_3-14.1x_4=15.5 \\ 2.4x_1-4.4x_2+3.5x_3+5.5x_4=2.5 \\ 5.4x_1+8.3x_2-7.4x_3-12.7x_4=8.6 \\ 6.3x_1-7.6x_2+1.34x_3+3.7x_4=12.1 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Гивенса. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 3.2x_1-2.5x_2+3.7x_3=6.5 \\ 0.5x_1+0.34x_2+1.7x_3=-0.24 \\ 1.6x_1+2.3x_2-1.5x_3=4.3 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 3.2x_1 - 2.5x_2 + 3.7x_3 = 6.5 \\ 0.5x_1 + 0.34x_2 + 1.7x_3 = -0.24 \\ 1.6x_1 + 2.3x_2 - 1.5x_3 = 4.3 \\ 3.6x_1 + 1.8x_2 - 4.7x_3 = 3.8 \end{cases}$$

Вариант 7.

1. Создать произвольную нижнетреугольную матрицу A 5 порядка (не унитреугольную), вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 14.4x_1 - 5.3x_2 + 14.3x_3 - 12.7x_4 = 14.4 \\ 23.4x_1 - 14.2x_2 - 5.4x_3 + 2.1x_4 = 6.6 \\ 6.3x_1 - 13.2x_2 - 6.5x_3 + 14.3x_4 = 9.4 \\ 5.6x_1 + 8.8x_2 - 6.7x_3 - 23.8x_4 = 7.3 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Хаусхолдера. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом Зейделя с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 1.24x_1 + 0.62x_2 - 0.95x_3 = 1.43 \\ 2.15x_1 - 1.18x_2 + 0.57x_3 = 2.43 \\ 1.72x_1 - 0.83x_2 + 1.57x_3 = 3.88 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 14.4\,x_1 - 5.3\,x_2 + 14.3\,x_3 = 14.4 \\ 23.4\,x_1 - 14.2\,x_2 - 5.4\,x_3 = 6.6 \\ 6.3\,x_1 - 13.2\,x_2 - 6.5\,x_3 = 9.4 \\ 5.6\,x_1 + 8.8\,x_2 - 6.7\,x_3 = 7.3 \end{cases}$$

Вариант 8.

1. Создать произвольную верхнюю унитреугольную матрицу A 5 порядка, вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 1.7\,x_1 + 10\,x_2 - 1.3\,x_3 + 2.1\,x_4 = 3.3 \\ 3.1\,x_1 + 1.7\,x_2 - 2.1\,x_3 + 5.4\,x_4 = 2.1 \\ 3.3\,x_1 - 7.7\,x_2 + 4.4\,x_3 - 5.1\,x_4 = 1.9 \\ 10\,x_1 - 20.1\,x_2 + 24\,x_3 + 1.7\,x_4 = 1.8 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Грама-Шмидта. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 3.6\,x_1 + 1.8\,x_2 - 4.7\,x_3 = 3.8 \\ 2.7\,x_1 - 3.6\,x_2 + 1.9\,x_3 = 0.4 \\ 1.5\,x_1 + 4.5\,x_2 + 3.3\,x_3 = -1.6 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 3.6x_1 + 1.8x_2 - 4.7x_3 = 3.8 \\ 2.7x_1 - 3.6x_2 + 1.9x_3 = 0.4 \\ 1.5x_1 + 4.5x_2 + 3.3x_3 = -1.6 \\ 5.6x_1 + 8.8x_2 - 6.7x_3 = 7.3 \end{cases}$$

Вариант 9.

1. Создать произвольную нижнюю унитреугольную матрицу A 7 порядка, вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 1.7x_1 - 1.8x_2 + 1.9x_3 - 57.4x_4 = 10 \\ 1.1x_1 - 4.3x_2 + 1.5x_3 - 1.7x_4 = 19 \\ 1.2x_1 + 1.4x_2 + 1.6x_3 + 1.8x_4 = 20 \\ 7.1x_1 - 1.3x_2 - 4.1x_3 + 5.2x_4 = 10 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Гивенса. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы (можно пользоваться модулями PrettyTable или Pandas).

$$\begin{cases} 0.43x_1 + 1.24x_2 - 0.58x_3 = 2.71 \\ 0.74x_1 + 0.83x_2 + 1.17x_3 = 1.26 \\ 1.43x_1 - 1.58x_2 + 0.83x_3 = 1.03 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 1.7x_1 - 1.8x_2 + 1.9x_3 = 10 \\ 1.1x_1 - 4.3x_2 + 1.5x_3 = 19 \\ 1.2x_1 + 1.4x_2 + 1.6x_3 = 20 \\ 7.1x_1 - 1.3x_2 - 4.1x_3 = 10 \end{cases}$$

Вариант 10.

1. Создать произвольную верхнетреугольную матрицу A 4 порядка (не унитреугольную), вектор B произвольный. Решить систему AX = B.

2. Решить систему, используя LU разложение

$$\begin{cases} 6.1x_1 + 6.2x_2 - 6.3x_3 + 6.4x_4 = 6.5 \\ 1.1x_1 - 1.5x_2 + 2.2x_3 - 3.8x_4 = 4.2 \\ 5.1x_1 - 5x_2 + 4.9x_3 - 4.8x_4 = 4.7 \\ 1.8x_1 + 1.9x_2 + 2x_3 - 2.1x_4 = 2.2 \end{cases}$$

3. Решить систему из пункта 2 с помощью QR разложения матрицы A. QR разложение найти методом Хаусхолдера. Проверить полученное решение непосредственной подстановкой в исходную систему, а также методом np.solve.

4. Решить систему методом простых итераций с точностью до 10^{-3}. Проверить выполнение достаточного условия сходимости. Если условие не выполняется, в программе выполнить эквивалентные преобразования системы, после этого привести к удобному для итераций виду. Оформить итерации в виде таблицы.

$$\begin{cases} 2.7x_1 + 0.9x_2 - 1.5x_3 = 3.5 \\ 4.5x_1 - 2.8x_2 + 6.7x_3 = 2.6 \\ 5.1x_1 + 3.7x_2 - 1.4x_3 = -0.14 \end{cases}$$

5. Найти псевдорешение системы:

$$\begin{cases} 2.7\,x_1 + 0.9\,x_2 - 1.5\,x_3 = 3.5 \\ 4.5\,x_1 - 2.8\,x_2 + 6.7\,x_3 = 2.6 \\ 5.1\,x_1 + 3.7\,x_2 - 1.4\,x_3 = -0.14 \\ 7.1\,x_1 - 1.3\,x_2 - 4.1\,x_3 = 10 \end{cases}$$

4 ИНДИВИДУАЛЬНЫЕ ДОМАШНИЕ ЗАДАНИЯ

В пункте a) определить, какое равенство точнее. В пункте b) округлить сомнительные цифры числа, оставив верные знаки. Определить абсолютную погрешность результата. В пункте c) найти предельные абсолютную и относительную погрешности приближенного числа, все цифры которого по умолчанию верные.

Варианты индивидуальных заданий.

1. *a*) $\frac{12}{17}=0{,}7059$; $\sqrt{53}=7{,}28$; *b*) 23,3745, δ=0,27% ; *c*) 0,645.

2. *a*) $\frac{12}{7}=1{,}7143$; $\sqrt{51}=7{,}14$; *b*) 13,3785, Δ=0,0027 ; *c*) 4,625.

3. *a*) $\frac{32}{18}=1{,}777$; $\sqrt{42}=6{,}48$; *b*) 0,3374, δ=0,17% ; *c*) 0,341.

4. *a*) $\frac{23}{9}=2{,}56$; $\sqrt{41}=6{,}403$; *b*) 4,351, Δ=0,007 ; *c*) 8,125.

5. *a*) $\frac{31}{13}=2{,}38$; $\sqrt{47}=6{,}86$; *b*) 10,334, δ=0,18% ; *c*) 0,034.

6. *a*) $\frac{21}{17}=1{,}235$; $\sqrt{61}=7{,}81$; *b*) 42,3051, Δ=0,03 ; *c*) 82,25.

7. *a*) $\frac{2}{7}=0{,}286$; $\sqrt{13}=3{,}605$; *b*) 23,375, δ=0,02% ; *c*) 67,445.

8. *a*) $\frac{4}{17}=0{,}235$; $\sqrt{105}=10{,}25$; *b*) 1,7051, Δ=0,013 ; *c*) 182,25.

9. *a*) $\frac{7}{15}=0{,}467$; $\sqrt{30}=5{,}48$; *b*) 7,375, δ=0,12% ; *c*) 17,1539.

10. *a*) $\frac{7}{9}=0{,}777$; $\sqrt{68}=8{,}246$; *b*) 2,1051, $\Delta=0{,}005$; *c*) 182,125.

11. *a*) $\frac{7}{8}=0{,}875$; $\sqrt{33}=5{,}7446$; *b*) 7,75, $\delta=0{,}2\%$; *c*) 7,153.

12. *a*) $\frac{2}{21}=0{,}095$; $\sqrt{22}=4{,}69$; *b*) 4,3851, $\Delta=0{,}007$; *c*) 572,95.

13. *a*) $\frac{5}{3}=1{,}667$; $\sqrt{14}=3{,}74$; *b*) 0,8374, $\delta=0{,}01\%$; *c*) 8,179.

14. *a*) $\frac{21}{29}=0{,}724$; $\sqrt{27}=5{,}20$; *b*) 0,3851, $\Delta=0{,}08$; *c*) 0,5729.

15. *a*) $\frac{61}{13}=4{,}692$; $\sqrt{44}=6{,}63$; *b*) 10,334, $\delta=0{,}18\%$; *c*) 0,034.

16. *a*) $\frac{6}{11}=0{,}545$; $\sqrt{83}=9{,}11$; *b*) 7,5121, $\Delta=0{,}008$; *c*) 2,5749.

17. *a*) $\frac{17}{19}=0{,}895$; $\sqrt{94}=9{,}6953$; *b*) 14,451, $\delta=0{,}11\%$; *c*) 0,57.

18. *a*) $\frac{6}{29}=0{,}2068$; $\sqrt{17}=4{,}123$; *b*) 16,851, $\Delta=0{,}004$; *c*) 0,579.

19. *a*) $\frac{4}{19}=0{,}2105$; $\sqrt{7}=2{,}6458$; *b*) 8,453, $\delta=0{,}21\%$; *c*) 0,23.

20. *a*) $\frac{26}{15}=1{,}733$; $\sqrt{57}=7{,}5498$; *b*) 17,152, $\Delta=0{,}0023$; *c*) 4,24.

21. *a*) $\frac{4}{19}=0{,}2105$; $\sqrt{7}=2{,}6458$; *b*) 38,451, $\delta=0{,}01\%$; *c*) 3,38.

22. *a*) $\frac{26}{25}=1{,}04$; $\sqrt{75}=8{,}6603$; *b*) 9,865, $\Delta=0{,}03$; *c*) 24,2174.

23. *a*) $\frac{61}{23}=2{,}6522$ *;* $\sqrt{84}=9{,}165$; *b*) 23,332, δ=0,23% ; *c*) 3,04.

24. *a*) $\frac{26}{31}=0{,}8387$ *;* $\sqrt{15}=3{,}873$; *b*) 9,65, Δ=0,08 ; *c*) 2,1741.

25. *a*) $\frac{1}{23}=0{,}0435$ *;* $\sqrt{43}=6{,}557$; *b*) 2,332, δ=0,02% ; *c*) 34,24.

26. *a*) $\frac{23}{51}=0{,}4509$ *;* $\sqrt{5}=2{,}236$; *b*) 0,861, Δ=0,04 ; *c*) 4,2179.

27. *a*) $\frac{1}{13}=0{,}08$ *;* $\sqrt{3}=1{,}732$; *b*) 2,32, δ=0,02% ; *c*) 14,23.

28. *a*) $\frac{3}{52}=0{,}0577$ *;* $\sqrt{55}=7{,}416$; *b*) 1,2321, Δ=0,007 ; *c*) 4,321.

29. *a*) $\frac{7}{13}=0{,}54$ *;* $\sqrt{73}=8{,}544$; *b*) 12,002, δ=0,06% ; *c*) 4,123.

30. *a*) $\frac{7}{52}=0{,}13$ *;* $\sqrt{48}=6{,}928$; *b*) 41,221, Δ=0,09 ; *c*) 54,345.

4.1 Пример решения

a) $\frac{9}{19}=0{,}474$ *;* $\sqrt{103}=10{,}149$.

Найдем значения этих выражений с большим числом десятичных знаков:

$a = 9/19 = 0.47368...$, $b = \sqrt{103} = 10.14889...$

Вычислим предельные абсолютные погрешности, округляя их с избытком:

$\Delta a=|0{,}47368-0{,}474|=3{,}2\cdot 10^{-4}\leqslant 0{,}0004$

$$\Delta b = |10{,}14889 - 10{,}149| = 1{,}1 \cdot 10^{-4} \leqslant 0{,}0002$$

Предельные относительные погрешности составляют

$$\delta a = \frac{\Delta a}{|a|} = \frac{0{,}0004}{0{,}474} \approx 8 \cdot 10^{-4} = 0{,}08\,\%$$

$$\delta b = \frac{\Delta b}{|b|} = \frac{0{,}0002}{10{,}149} \approx 2 \cdot 10^{-5} = 0{,}002\,\%$$

Поскольку δb меньше, чем δa, то второе равенство является более точным.

b) Пусть $c = 72{,}354$, $\Delta c = 0{,}021$

Предельная абсолютная погрешность $\Delta c = 0{,}021 < 0{,}1$, следовательно число *c* придется округлить до десятых и окончательно, только с верными цифрами, его запишем в виде 72.4.

Пусть $d = 5{,}272$, $\delta d = 0{,}1\,\%$

Поскольку $\Delta d = |d| \cdot \delta d = 5{,}272 \cdot 0{,}001 = 0{,}005272 < 0{,}01$, то число d придется округлить до сотых и окончательно, только с верными цифрами, его запишем в виде 5.27.

c) Пусть приближенное число *f* = 14,278 . Поскольку все его цифры верные, то предельная абсолютная погрешность $\Delta f \leqslant 0{,}001$. Тогда предельная относительная погрешность

$$\delta f = \frac{\Delta f}{|f|} = \frac{0{,}001}{14{,}278} \approx 7 \cdot 10^{-5} \approx 10^{-4} = 0{,}01\,\%$$

Список литературы

1. Амосов, А.А. Вычислительные методы: учебное пособие / А.А. Амосов, Ю.А. Дубинский, Н.В. Копченова. – 4-е изд., стер. – Санкт-Петербург: Лань, 2022. – 672 с.
2. Бахвалов, Н.С. Численные методы: учебник / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – 9-е изд. – Москва: Лаборатория знаний, 2020. – 636 с.
3. Вержбицкий, В.М. Численные методы: Линейная алгебра и нелинейные уравнения: учебное пособие для студентов мат. и инж. специальностей вузов / В.М. Вержбицкий. – Москва: Высш. шк., 2000. – 265 с.
4. Вержбицкий, В.М. Основы численных методов: учебник для студентов высших учебных заведений, обучающихся по направлению подготовки дипломированных специалистов «Прикладная математика» / В.М. Вержбицкий. – 3-е изд., стер. – Москва: Высш. шк., 2009. – 839 с.
5. Воеводин, В.В. Вычислительные основы линейной алгебры / В.В. Воеводин. – Москва: Наука, 1977. – 303 с.
6. Воеводин, В.В. Матрицы и вычисления / В.В. Воеводин, Ю.А. Кузнецов. – Москва: Наука, 1984. – 320 с.
7. Волков, Е.А. Численные методы / Е.А. Волков. – Москва: Наука, 1982. – 256 с.
8. Воробьева, Г.Н. Практикум по численным методам / Г.Н. Воробьева, А.Н. Данилова. – Москва: Высш. шк., 1979. – 184 с.
9. Голуб, Дж. Матричные вычисления / Дж. Голуб, Ч. Ван Лоун. – пер. с англ. – Москва: Мир, 1999. – 548 с.
10. Деммель, Дж. Вычислительная линейная алгебра. Теория и приложения / Дж. Деммель. – Москва: Мир, 2001. – 435 с.
11. Демидович, Б.П. Основы вычислительной математики: учебное пособие / Б.П. Демидович, И.А. Марон. – 8-е изд., стер. – Санкт-Петербург: Лань, 2022. – 672 с.

12. Зенков, А.В. Численные методы: учебное пособие / А.В. Зенков; научный редактор В.В. Плещев. – Екатеринбург: УрФУ, 2016. – 124 с.

13. Калиткин, Н.Н. Численные методы / Н.Н. Калиткин. – Москва: Наука, 1978. – 512 с.

14. Копченова, Н.В. Вычислительная математика в примерах и задачах: учебное пособие для вузов / Н.В. Копченова, И.А. Марон. – 5-е изд., стер. – Санкт-Петербург: Лань, 2021. – 368 с.

15. Лапчик, М.П. Численные методы: учебное пособие для студентов вузов / М.П. Лапчик, М.И. Рагулина, Е.К. Хеннер. – Москва: Академия, 2005. – 384 с.

16. Минькова, Р.М. Методы вычислительной математики / Р.М. Минькова, Р.А. Вайсбурд. – Свердловск: Издательство УПИ им. С.М. Кирова, 1981. – 88 с.

17. Уилкинсон, Дж.Х. Алгебраическая проблема собственных значений / Дж.Х. Уилкинсон. – Москва: Наука, 1970. – 564 с.

18. Фаддеев, Д.К. Вычислительные методы линейной алгебры: учебник / Д.К. Фаддеев, В.Н. Фаддеева. – 4-е изд., стер. – Санкт-Петербург: Лань, 2022. – 736 с.

19. Фаддеев, Д.К. Лекции по алгебре / Д.К. Фаддеев. – Москва: Наука, 1984. – 416 с.

20. Форсайт, Дж. Машинные методы математических вычислений: учебник / Дж. Форсайт, М. Малькольм, К. Моулер. – 1-е изд. – Москва: Мир, 1980. – 276 с.

21. Численные методы / Н.И. Данилина [и др.]. – Москва: Высш. шк., 1976. – 368 с.

22. Хемминг, Р.И. Численные методы для научных работников и инженеров / Р.И. Хемминг. – Москва: Мир, 1972. – 399 с.

Учебное издание

Коновалова Елена Игоревна,
Яблокова Людмила Вениаминовна

ЧИСЛЕННЫЕ МЕТОДЫ ЛИНЕЙНОЙ АЛГЕБРЫ

Учебное пособие

Редакционно-издательская обработка
издательства Самарского университета

Подписано в печать 26.12.2022. Формат 60x84 1/16.
Бумага офсетная. Печ. л. 9,5.
Тираж 120 экз. (1-й з-д 1-25). Заказ . Арт. – 43(Р2УП)/2022.

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»
(САМАРСКИЙ УНИВЕРСИТЕТ)
443086, САМАРА, МОСКОВСКОЕ ШОССЕ, 34.

Издательство Самарского университета.
443086, Самара, Московское шоссе, 34.