Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

С.П. Куксенко, Т.Р. Газизов

Итерационные методы решения системы линейных алгебраических уравнений с плотной матрицей

Томск
2007

УДК 519.612
ББК 22.193.1
К 898

Монография подготовлена и издана за счет средств национального проекта «Образование»

К 898

Куксенко С.П., Газизов Т.Р.
Итерационные методы решения системы линейных алгебраических уравнений с плотной матрицей. – Томск: Томский государственный университет, 2007. – 208 с.

ISBN 5-94621-226-5

Рассмотрен учет погрешностей приближенных вычислений. Изложены прямые и итерационные методы решения системы линейных алгебраических уравнений (СЛАУ). Приведены алгоритмы итерационных методов с предобусловливанием для эффективного решения СЛАУ с плотной матрицей большого порядка. Представлены результаты вычислительных экспериментов по решению СЛАУ итерационными методами при использовании различных видов алгебраической предфильтрации, способов предобусловливания и их параметров. Приведены исходные коды программ на C++.

Для студентов и аспирантов физико-математических, экономических и инженерных специальностей.

УДК 519.612
ББК 22.193.1

Рецензент: д.ф-м.н., проф. А.Г. Дмитренко,
Томский государственный университет

ISBN 5-94621-226-5

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Решение системы линейных алгебраических уравнений (СЛАУ) имеет большое значение, поскольку к нему сводится решение широкого круга сложных практических задач. В линейной алгебре эту задачу называют первой основной задачей. Так, около 75% всех расчетных математических задач приходится на решение СЛАУ. Хотя эта задача 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 известно, значительная часть численных методов решения различных (в особенности – нелинейных) задач включает в себя решение СЛАУ как элементарный шаг соответствующего алгоритма.

Необходимость решения СЛАУ возникает при решении многомерных анизотропных краевых задач, в задачах вычислительной гидродинамики, в теории электрических цепей, при решении уравнений балансов и сохранения в механике, гидравлике и т.п. В геомеханике матрица СЛАУ имеет чрезмерно большие размеры и является плохо обусловленной. Поэтому обычные методы решения СЛАУ здесь оказываются неэффективными. Задача нахождения устойчивых приближенных решений СЛАУ является определяющей задачей для гравиметрии и магнитометрии. Необходимость решения трехдиагональных СЛАУ возникает и в физике (оптика, теория теплопроводности, газовая динамика и др.), и в математике (теория разностных схем, проекционные и вариационные методы). Также проблема решения СЛАУ существует в задачах управления и контроля, которые предъявляют высокие требования к скорости получения результатов, пусть даже приближенных. Среди них можно выделить класс задач оценки и предсказания критических ситуаций, связанных, например, с измерением температуры и вычислением плотности теплового потока на поверхности спускаемого летательного аппарата и др.

Задача решения СЛАУ имеет незапамятную историю. Так, самый известный и уже ставший классическим метод исключения, развиваемый и изучаемый даже в наши дни, был предложен К. Гауссом в 1849 г. Однако еще до н.э. в Китае были изданы «Девять книг о математическом искусстве», где этот алгоритм был изложен в характерной для своего времени «натуральной» форме, но фактически с использованием матричных преобразований.

Становление современных вычислительных методов линейной алгебры можно считать состоявшимся после выхода в 1963 г. книги Фадеевых (Фадеев Д.К., Фадеева В.Н. Вычислительные методы линейной алгебры).

Необходимо подчеркнуть, что актуальные проблемы вычислительной алгебры имеют фундаментальный характер не только потому, что текущая компьютеризация различных областей знаний в значительной степени сводится к векторно-матричным процедурам. Изучение матриц, являющихся операторами в простейших конечномерных пространствах, позволяет обнаружить наиболее глубокие и тонкие свойства математических объектов, имеющие свое значение для функционального анализа, теории аппроксимации, дифференциальных уравнений и т.д.

В настоящее время имеется значительное число учебников и монографий, посвященных вычислительным методам. Однако в них методам решения СЛАУ уделено, как правило, недостаточное внимание и не отражена вся совокупность этих методов. Кроме того, большинство этих книг ориентировано на студентов-математиков или на специалистов по вычислительной математике. В то же время практически отсутствует отечественная учебная литература, в которой доступным для студента технического вуза языком были бы изложены основы методов решения линейных систем, применяемых для решения практических задач. Особенно острой, по мнению авторов, является потребность в книге, которая давала бы представление о реально используемых в вычислительной практике алгоритмах. Данная работа призвана в определенной степени восполнить этот пробел. Кроме того, книга содержит результаты вычислительных экспериментов, проведенных авторами на практических задачах, а также листинг работающих и используемых кодов на C++. Таким образом, работа включает не только уникальный обзор, но и органично дополняющие его оригинальные материалы с конечными результатами.

Дадим краткое изложение содержания книги.

В гл. 1 наряду с введением в элементарную теорию погрешностей содержится изложение основных особенностей машинной арифметики. Понимание этих особенностей необходимо тем, кто заинтересован в эффективном применении ЭВМ для решения прикладных задач. Данная глава основана на [1], поскольку эта работа включает все необходимые сведения для усвоения дальнейшего материала.

В гл. 2 рассмотрены прямые (точные) методы решения СЛАУ. Основное внимание уделяется методу Гаусса и его различным модификациям. Рассматриваются использование LU-разложения матриц для решения СЛАУ, метод Крамера, матричный метод, метод прогонки, метод Гаусса–Жордана, метод квадратного корня, методы вращений и отражений. Обсуждается алгоритм итерационного уточнения. Некоторые из этих методов используются в схемах итерационных методов, а также для повышения быстродействия итерационного процесса. Так, например, очень часто

LU-разложение используется при формировании матрицы предобусловливания.

В гл. 3 рассмотрены итерационные методы решения СЛАУ, начиная с классических (Якоби, Гаусса–Зейделя, релаксации и др.). Особое внимание уделено построению так называемых проекционных методов и, в частности, того их класса, который связан с проектированием на подпространства Крылова. Так, детально рассмотрены методы бисопряженных градиентов, квази-минимальных невязок, сопряженных градиентов и др.

В гл. 4. показано уменьшение времени решения СЛАУ итерационными методами за счет использования предфильтрации и предобусловливания. Также приведены результаты вычислительных экспериментов по сравнению способов предобусловливания и предфильтрации, проведенных авторами на примере вычисления токов в проводной антенне.

В приложении приведен листинг файлов, содержащих функции, позволяющие решать СЛАУ итерационными методами с несколькими способами предобусловливания и предфильтрации, а также прямыми методами. Данные файлы используются в модуле MATRIX системы TALGAT и показали свою работоспособность.

Монография может быть использована в высшем профессиональном образовании в качестве учебного пособия по естественнонаучным и специальным дисциплинам, связанным с вычислительной математикой, математическим моделированием, автоматизированным проектированием. Так, материалы монографии в течение нескольких лет используются авторами в Томском государственном университете систем управления и радиоэлектроники в образовательной программе высшего профессионального образования для студентов специальностей 201500 – «Бытовая радиоэлектронная аппаратура», 201400 – «Аудиовизуальная техника», 230700 – «Сервис» по дисциплинам: «Электромагнитная совместимость и безопасность радиоэлектронной аппаратуры»; «Основы компьютерного проектирования и моделирования РЭС»; «Системы автоматизированного проектирования в сервисе»; «Учебно-исследовательская работа», а также студентами и аспирантами в ходе научно-исследовательской работы, учебно-научного проектирования и группового проектного обучения по направлению «Электромагнитная совместимость».

Авторы выражают большую признательность за помощь и поддержку в этой работе А.О. Мелкозерову, Т.Т. Газизову, А.М. Заболоцкому, И.С. Костареву, С.Т. Сивцеву и за ряд ценных замечаний А.Г. Дмитренко и Н.Д. Малютину. Авторы благодарят ректорат ТУСУРа за помощь в издании этой монографии.

1. УЧЕТ ПОГРЕШНОСТЕЙ ПРИБЛИЖЕННЫХ ВЫЧИСЛЕНИЙ

1.1. Источники и классификация погрешностей результата численного решения задачи

Для правильного понимания подходов и критериев, используемых при решении прикладной задачи с применением компьютера, очень важно с самого начала признавать, что получить точное значение решения практически невозможно и не в этом цель вычислений. Получаемое на компьютере решение x^* почти всегда (за исключением некоторых весьма специальных случаев) содержит погрешность, т.е. является приближенным. Невозможность получения точного решения следует уже из ограниченной разрядности компьютера.

Наличие погрешности решения обусловлено рядом причин. Перечислим их.

Математическая модель является лишь приближенным описанием реального процесса. Характеристики процесса, вычисленные в рамках принятой модели, заведомо отличаются от истинных, причем их погрешность зависит от степени адекватности модели реальному процессу.

Исходные данные, как правило, содержат погрешности, поскольку они либо получаются в результате экспериментов (измерений), либо являются результатом решения некоторых вспомогательных задач.

Применяемые для решения задачи методы в большинстве случаев являются приближенными. Найти решение возникающей на практике задачи в виде конечной формулы возможно только в отдельных, очень упрощенных ситуациях.

При вводе исходных данных в компьютер, выполнении арифметических операций и выводе результатов на печать производятся округления.

Пусть x – точное значение величины, вычисление которого является целью поставленной задачи. Соответствующая первым двум из указанных причин погрешность $\delta_н x$ называется неустранимой погрешностью. Такое название вызвано тем, что принятие математической модели и задание исходных данных вносит в решение погрешность, которая не может быть устранена далее. Единственный способ уменьшить эту погрешность – перейти к более точной математической модели и задать более точные исходные данные.

Погрешность $\delta_м x$, источником которой является метод решения задачи, называется погрешностью метода, а погрешность $\delta_в x$ из-за округлений при вводе, выводе и вычислениях – вычислительной погрешностью.

Таким образом, полная погрешность результата решения задачи на компьютере $\delta x = x - x^*$ складывается из трех составляющих: неустранимой погрешности, погрешности метода и вычислительной погрешности, т.е. $\delta x = \delta_н x + \delta_м x + \delta_в x$.

Будем далее исходить из предположения, что математическая модель фиксирована и входные данные задаются извне, так что повлиять на значение величины $\delta_н x$ в процессе решения задачи действительно нельзя. Однако это совсем не означает, что предварительные оценки значения неустранимой погрешности не нужны. Достоверная информация о порядке величины $\delta_н x$ позволяет осознанно выбрать метод решения задачи и разумно задать его точность. Желательно, чтобы погрешность метода была в 2–10 раз меньше неустранимой погрешности. Большее значение $\delta_м x$ ощутимо снижает точность результата, меньшее – обычно требует увеличения затрат на вычисления, практически уже не влияя на значение полной погрешности. Иногда характер использования результата таков, что вполне допустимо, чтобы погрешность $\delta_м x$ была сравнима с $\delta_н x$ или даже несколько превышала ее.

Значение вычислительной погрешности (при фиксированных модели, входных данных и методе решения) в основном определяется характеристиками используемого компьютера. Желательно, чтобы погрешность $\delta_в x$ была бы хоть на порядок меньше погрешности метода и совсем не желательна ситуация, когда она существенно ее превышает.

Умение анализировать погрешности при решении прикладной задачи и соблюдать между ними разумный компромисс позволяет существенно экономить используемые ресурсы и требует высокой квалификации.

1.2. Приближенные числа. Абсолютная и относительная погрешности

В разделе 1.1 было отмечено, что числа, получаемые при решении прикладных задач на компьютере, как правило, являются приближенными. Следовательно, вопрос о точности результатов, т.е. о мере их уклонения от истинных значений, в теории и практике методов вычислений приобретает особое значение. Начнем его рассмотрение с введения основных понятий элементарной теории погрешностей.

1.2.1. *Абсолютная и относительная погрешности*

Пусть a – точное (вообще говоря, неизвестное) значение некоторой величины, a^* – известное приближенное значение той же величины (*приближенное число*). Ошибкой или погрешностью приближенного числа a^* называют разность $a - a^*$ между точным и приближенным значениями.

Простейшей количественной мерой погрешности является абсолютная погрешность

$$\Delta(a^*) = |a - a^*|. \tag{1.1}$$

Однако по значению абсолютной погрешности далеко не всегда можно сделать правильное заключение о качестве приближения. Действительно, если $\Delta(a^*) = 0.1$, то следует ли считать погрешность большой или нужно признать ее малой? Ответ существенным образом зависит от принятых единиц измерения и масштабов величин. Если $a \approx 0.3$ то, скорее всего, точность приближения невелика; если же $a \approx 3 \cdot 10^8$, то следует признать точность очень высокой. Таким образом, естественно соотнести погрешность величины и ее значение, для чего вводится понятие относительной погрешности (при $a \neq 0$)

$$\delta(a^*) = |a - a^*| / |a| = \Delta(a^*) / |a|. \tag{1.2}$$

Использование относительных погрешностей удобно, в частности, тем, что они не зависят от масштабов величин и единиц измерения. Заметим, что для приведенного примера $\delta(a^*) \approx 0.33 = 33\%$ в первом случае и $\delta(a^*) \approx 0.33 \cdot 10^{-9} = 0.33 \cdot 10^{-7}\%$ во втором.

Так как значение a неизвестно, то непосредственное вычисление величин $\Delta(a^*)$ и $\delta(a^*)$ по формулам (1.1) и (1.2) невозможно. Более реальная и часто поддающаяся решению задача состоит в получении оценок погрешности вида

$$|a - a^*| \le \Delta'(a^*), \tag{1.3}$$

$$|a - a^*| / |a| \le \delta'(a^*), \tag{1.4}$$

где $\Delta'(a^*)$ и $\delta'(a^*)$ – величины, которые будем называть верхними границами (или границами) абсолютной и относительной погрешностей.

Если величина $\Delta'(a^*)$ известна, то неравенство (1.4) будет выполнено, если положить

$$\delta'(a^*) = \Delta'(a^*) / |a|. \tag{1.5}$$

Точно так же, если величина $\delta'(a^*)$ известна, то следует положить

$$\Delta'(a^*) = |a|\delta'(a^*). \tag{1.6}$$

Поскольку значение a неизвестно, при практическом применении формулы (1.5), (1.6) заменяют приближенными равенствами

$$\delta'(a^*) \approx \Delta'(a^*) / |a^*|, \ \Delta'(a^*) \approx |a^*|\delta'(a^*). \tag{1.7}$$

В литературе по методам вычислений широко используется термин «точность». Принято говорить о точности входных данных и решения, о повышении и снижении точности вычислений и т.д. Точность в качественных рассуждениях обычно выступает как противоположность погрешности, хотя для количественного их измерения используются одни и те же характеристики (например, абсолютная и относительная погрешно-

сти). Точное значение величины – это значение, не содержащее погрешности. Повышение точности воспринимается как уменьшение погрешности, а снижение точности – как увеличение погрешности. Часто используемая фраза «требуется найти решение с заданной точностью ε» означает, что ставится задача о нахождении приближенного решения, принятая мера погрешности которого не превышает заданного значения ε. Вообще говоря, следовало бы говорить об абсолютной точности и относительной точности, но часто этого не делают, считая, что из контекста ясно, как измеряется погрешность.

1.2.2. *Правила записи приближенных чисел*

Пусть приближенное число a^* задано в виде конечной десятичной дроби:

$$a^* = \alpha_n\alpha_{n-1}\ldots\alpha_0.\beta_1\beta_2\ldots\beta_m.$$

Значащими цифрами числа a^* называют все цифры в его записи, начиная с первой ненулевой слева.

Пример 1.1. У чисел $a^* = 0.0\underline{103}$ и $a^* = 0.0\underline{103000}$ значащие цифры подчеркнуты. Первое число имеет три, а второе – шесть значащих цифр.

Значащую цифру числа a^* называют верной, если абсолютная погрешность числа не превосходит единицы разряда, соответствующего этой цифре.

Пример 1.2. Если $\Delta'(a^*) = 2\cdot10^{-6}$, то число $a^* = 0.0\underline{10300}0$ имеет четыре верные значащие цифры (они подчеркнуты).

Следует отметить, что широко распространенной ошибкой при записи приближенных чисел является отбрасывание последних значащих нулей (даже если они представляют собой верные цифры). Верная цифра приближенного числа, вообще говоря, не обязана совпадать с соответствующей цифрой в записи точного числа. Таким образом, термин «верная цифра» не следует понимать буквально (см. пример 1.3).

Пример 1.3. Пусть $a = 1.00000$, $a^* = 0.\underline{99999}$. Тогда $\Delta'(a^*) = 0.00001$ и у числа a^* все подчеркнутые цифры – верные, хотя и не совпадают с соответствующими цифрами числа a.

Количество верных значащих цифр числа тесно связано со значением его относительной погрешности. Приведенные ниже утверждения позво-

ляют в дальнейшем связывать точность числа с количеством его верных значащих цифр и трактовать потерю точности как потерю верных цифр.

Утверждение 1.1.

1. Если число a^* содержит N верных значащих цифр, то справедливо
$$\delta'(a^*) \le (10^{N-1} - 1)^{-1} \approx 10^{-N+1}.$$
2. Для того чтобы число a^* содержало N верных значащих цифр, достаточно чтобы было выполнено неравенство
$$\delta'(a^*) \le (10^N + 1)^{-1} \approx 10^{-N}.$$
3. Если число a^* имеет ровно N верных значащих цифр, то
$$10^{-N-1} \le \delta'(a^*) \le 10^{-N+1}$$
и таким образом $\delta'(a^*) \approx 10^{-N}$.

Пример 1.4. Что можно сказать об относительной погрешности числа a^*, если известно, что оно содержит три верные значащие цифры?

В силу утверждения 1 имеем $\delta'(a^*) \le 10^{-2} = 1\%$.

Пример 1.5. С какой относительной точностью следует найти число a^*, чтобы верными оказались шесть его значащих цифр?

Из утверждения 2 следует, что достаточно найти a^* с относительной точностью $\varepsilon \approx 10^{-6}$.

Заметим, что границы абсолютной и относительной погрешностей принято записывать с одной или двумя значащими цифрами. Большая точность в записи этих величин, как правило, не имеет смысла, так как обычно они являются довольно грубыми оценками истинных значений погрешностей, и, кроме того, для практического использования часто бывает достаточно знать только их порядок.

Пример 1.6. Информация о погрешности вида $\delta'(a^*) \approx 0.288754 \cdot 10^{-5}$ практически равноценна информации $\delta'(a^*) \approx 3 \cdot 10^{-6}$, причем последняя вызывает больше доверия. Скорее всего, вполне удовлетворительной в данном случае является запись $\delta'(a^*) \approx 10^{-6}$.

Неравенство (1.3) эквивалентно двойному неравенству
$$a^* - \Delta'(a^*) \le a \le a^* + \Delta'(a^*)$$
и поэтому тот факт, что число a^* является приближенным значением числа a с верхней границей абсолютной погрешности $\Delta'(a^*)$ (с абсолютной точностью $\varepsilon = \Delta'(a^*)$), принято записывать в виде

$$a = a^* \pm \Delta'(a^*).$$

Как правило, числа a^* и $\Delta'(a^*)$ указываются с одинаковым числом цифр после десятичной точки.

Пример 1.7. Пусть для числа a известны приближенное значение $a^* = 1.648$ и граница абсолютной погрешности $\Delta'(a^*) = 0.002832$. Тогда можно записать $a = 1.648 \pm 0.003$. Записи вида $a = 1.648 \pm 0.002832$ или $a = 1.648 \pm 0.1$ являются неестественными.

Из равенства (1.4) следует, что значение a заключено примерно между $a^*(1 - \delta'(a^*))$ и $a^*(1 + \delta'(a^*))$. Поэтому тот факт, что число a^* является приближенным значением числа a с границей относительной погрешности $\delta'(a^*)$ (с относительной точностью $\varepsilon = \delta'(a^*)$), принято записывать в виде $a = a^*(1 \pm \delta'(a^*))$.

Пример 1.8. Оценим точность часто используемого в простейших расчетах приближения $\pi^* = 3.14$ к числу π. Поскольку $\pi = 3.14159$, то $\pi - \pi^* = 0.00159$, следовательно, можно принять $\Delta'(\pi^*) = 0.0016$ и $\delta'(\pi^*) \approx$ $\approx 0.0016 / 3.14 \approx 0.00051 = 0.051\%$. Итак, $\pi = 3.14(1 \pm 0.051\%)$.

Если число a^* приводится в качестве результата без указания значения погрешности, то принято считать, что все его значащие цифры являются верными. Начинающий пользователь часто слишком доверяет выводимым из компьютера цифрам, предполагая, что вычислительная машина придерживается того же соглашения. Однако это совсем не так: число может быть выведено с таким количеством значащих цифр, сколько потребует программист заданием соответствующего формата. Как правило, среди этих цифр только небольшое число первых окажутся верными, а возможно верных цифр нет совсем. Анализировать результаты вычислений и определять степень их достоверности совсем не просто.

1.2.3. *Округление*

Часто возникает необходимость в округлении числа a, т.е. замене его другим числом a^* с меньшим числом значащих цифр. Возникающая при такой замене погрешность называется погрешностью округления.

Есть несколько способов округления числа до n значащих цифр. Простейший, усечение, состоит в отбрасывании всех цифр, расположенных справа от n-й значащей цифры. Более предпочтительным является округление по дополнению. Если первая слева из отбрасываемых цифр меньше

5, то сохраняемые цифры остаются без изменения. Если же она больше либо равна 5, то в младший сохраняемый разряд добавляется единица.

Абсолютное значение погрешности при округлении по дополнению не превышает половины единицы разряда, соответствующего последней оставляемой цифре, а при усечении – единицы того же разряда.

Пример 1.9. При округлении числа $a = 1.72631$ усечением до трех значащих цифр получится число $a^* = 1.72$, а при округлении по дополнению – число $a^* = 1.73$.

Границы абсолютной и относительной погрешностей принято всегда округлять в сторону увеличения.

Пример 1.10. Округление значений $\Delta'(a^*) = 0.003721$ и $\delta'(a^*) = 0.0005427$ до двух значащих цифр дает значения $\Delta'(a^*) = 0.0038$ и $\delta'(a^*) = 0.00055$.

1.3. Погрешности арифметических операций над приближенными числами

Исследуем влияние погрешностей исходных данных на погрешность результатов арифметических операций. Пусть a^* и b^* – приближенные значения чисел a и b. Какова соответствующая им величина неустранимой погрешности результата?

Утверждение 1.2. Абсолютная погрешность алгебраической суммы (суммы или разности) не превосходит суммы абсолютных погрешностей слагаемых, т.е.

$$\Delta(a^* \pm b^*) \le \Delta(a^*) + \Delta(b^*), \tag{1.8}$$

поскольку

$$\Delta(a^* \pm b^*) = |(a \pm b) - (a^* \pm b^*)| = |(a - a^*) \pm (b - b^*)| \le \Delta(a^*) + \Delta(b^*).$$

В силу неравенства (1.8) естественно положить

$$\Delta'(a^* \pm b^*) \le \Delta'(a^*) + \Delta'(b^*). \tag{1.9}$$

Оценим относительную погрешность алгебраической суммы.

Утверждение 1.3. Пусть a и b – ненулевые числа одного знака. Тогда справедливы неравенства

$$\delta(a^* + b^*) \le \delta_{max},\ \delta(a^* - b^*) \le \nu\, \delta_{max}, \tag{1.10}$$

где $\delta_{max} = \max\{\delta(a^*), \delta(b^*)\}$, $\nu = |a + b| / |a - b|$.

Используя формулу (1.2) и неравенство (1.8), имеем

$$|a \pm b|\delta(a^* \pm b^*) = \Delta(a^* \pm b^*) \le \Delta(a^*) + \Delta(b^*) =$$
$$= |a|\delta(a^*) + |a \pm |b|\delta(b^*) \le (|a| + |b|)\delta_{\max} = |a + b|\delta_{\max},$$

откуда видна очевидность оценки (1.10).

В силу неравенства (1.10) естественно положить

$$\delta'(a^* + b^*) = \delta'_{\max},\ \delta'(a^* - b^*) = \nu\,\delta'_{\max}, \qquad (1.11)$$

где $\delta'_{\max} = \max\{\delta'(a^*), \delta'(b^*)\}$, $\nu = |a + b| / |a - b|$.

Первое из равенств (1.11) означает, что при суммировании чисел одного знака не происходит потери точности, если оценить точность в относительных единицах. Совсем иначе дело обстоит при вычитании чисел одного знака. Здесь граница относительной погрешности возрастает в $\nu > 1$ раз и возможна существенная потеря точности. Если числа a и b близки настолько, что $|a + b| >> |a - b|$, то $\nu >> 1$ и не исключена полная или почти полная потеря точности. Когда это происходит, говорят о том, что произошла катастрофическая потеря точности.

Пример 1.11. Пусть решается инженерная задача, в которой окончательный результат y вычисляется по формуле $y = 1 - x$ с помощью предварительно определяемого значения x. Предположим, что найденное приближение $x^* = 0.999997$ к значению x содержит шесть верных значащих цифр. Тогда $y^* = 1 - 0.999997 = 0.000003$ и в процессе вычисления оказались потерянными пять верных цифр. Если же учесть, что $\delta(x^*) \approx$ $\approx 0.0001\%$, а $\delta(y^*) \approx 33\%$, то следует признать, что произошла катастрофическая потеря точности.

Подчеркнем, что здесь виновником «катастрофы» является не операция вычитания, а предложенный метод решения задачи, где окончательный результат получается с помощью вычитания двух близких чисел. Выполнение этой операции лишь делает очевидным то, что действительно полезная информация о значении y уже оказалась потерянной до вычитания. Если нет другого варианта расчета, то для получения приемлемого результата следовало бы предварительно вычислить x с существенно большим числом верных знаков, учитывая, что пять старших цифр при вычитании будут потеряны.

Итак, получаем следующий важный вывод. При построении численного метода решения задачи следует избегать вычитания близких чисел одного знака. Если же такое вычитание неизбежно, то следует вычислять аргументы с повышенной точностью, учитывая ее потерю примерно в $\nu = |a + b| / |a - b|$ раз.

Утверждение 1.4. Для относительных погрешностей произведения приближенных чисел верны оценки

$$\delta(a^* b^*) \le \delta(a^*) + \delta(b^*) + \delta(a^*)\delta(b^*), \quad (1.12)$$

$$\delta(a^* / b^*) \le \delta(a^*) + \delta(b^*) / (1 - \delta(b^*)), \quad (1.13)$$

в последней из которых считается, что $\delta(b^*) < 1$.

Для доказательства выполним следующие преобразования:

$$|ab|\delta(a^* b^*) = \Delta(a^* b^*) = |ab - a^* b^*| =$$
$$= |(a - a^*)b + (b - b^*)a - (a - a^*)(b - b^*)| \le$$
$$\le |b|\Delta(a^*) + |a|\Delta(b^*) + \Delta(a^*)\Delta(b^*) = |ab|(\delta(a^*) + \delta(b^*) + \delta(a^*)|ab|\delta(b^*)).$$

Разделив обе части полученного неравенства на $|ab|$, выведем оценку (1.12).

Для вывода второй оценки предварительно заметим, что

$$|b^*| = |b + (b^* - b)| \ge |b| - \Delta(b^*) = |b|(1 - \delta(b^*)).$$

Тогда

$$\delta(a^* / b^*) = |a / b - a^* / b^*| / |a / b| = |ab^* - ba^*| / |ab^*| =$$
$$= |a(b^* - b) + b(a - a^*)| / |ab^*| \le |a|\Delta(b^*) + |b|\Delta(a^*) / |ab|(1 - \delta(b^*)) =$$
$$= \delta(a^*) + \delta(b^*) / (1 - \delta(b^*)).$$

Если $\delta'(a^*) << 1$ и $\delta'(b^*) << 1$, то для оценки границ относительных погрешностей можно использовать следующие приближенные равенства:

$$\delta'(a^* b^*) \approx \delta'(a^*) + \delta'(b^*), \delta'(a^* / b^*) \approx \delta'(a^*) + \delta'(b^*). \quad (1.14)$$

Именно равенства (1.14) чаще всего и используют для практической оценки погрешности.

Итак, выполнение арифметических операций над приближенными числами, как правило, сопровождается потерей точности. Единственная операция, при которой потеря не происходит, – это сложение чисел одинакового знака. Наибольшая потеря точности может произойти при вычитании близких чисел одного знака.

1.4. Особенности машинной арифметики

Знание основных особенностей машинной арифметики необходимо для грамотного использования компьютеров при решении поставленных задач. Пользователь, не учитывающий эти особенности, вряд ли может рассчитывать на высокую точность и эффективность вычислений. Невнимание к ним часто приводит к неверным результатам. Подчеркнем, что в основе появления вычислительной погрешности лежит сам способ представления чисел в компьютере.

1.4.1. *Системы счисления*

Принятый способ записи чисел состоит в представлении их упорядоченным набором цифр. В привычной для нас десятичной позиционной системе счисления вещественное число x представляют последовательностью символов, которая начинается со знака (+ или −) и продолжается цепочкой десятичных цифр α_i и β_i, разделенных десятичной точкой:

$$x = \pm\alpha_n \ldots \alpha_1\alpha_0.\beta_1\beta_2 \ldots \beta_m \ldots \qquad (1.15)$$

Здесь каждой позиции (разряду), которую занимает цифра относительно десятичной точки, отвечает определенная степень числа 10. По существу, равенство (1.15) представляет собой принятое сокращение разложения числа x в сумму по степеням 10:

$$x = \pm(\alpha_n \cdot 10^n + \ldots + \alpha_1 \cdot 10^1 + \alpha_0 \cdot 10^0 + \beta_1 \cdot 10^{-1} + \beta_2 \cdot 10^{-2} + \ldots + \beta_m \cdot 10^{-m} + \ldots).$$

Пример 1.12. Запись $x = 20.5$ означает, что $x = 2 \cdot 10^1 + 0 \cdot 10^0 + 5 \cdot 10^{-1}$.

Для представления чисел в компьютере также используют позиционные системы счисления, однако основаниями систем служат, как правило, степени числа 2. Это вызвано способом хранения чисел в устройствах памяти компьютера, каждое из которых можно рассматривать как набор однотипных элементов, способных находиться только в одном из двух возможных устойчивых состояний – «включен» или «выключен». Эти состояния интерпретируются как нулевое или единичное значение двоичной цифры. Наиболее распространены системы счисления с основанием 2 (базисная двоичная система счисления), 8 и 16.

Игнорируя некоторые малосущественные детали, будем считать, что все вычислительные машины работают в двоичной системе счисления. В ней вещественное число x по-прежнему записывается в виде (1.15), однако α_i и β_i – уже двоичные цифры (0 или 1). В этом случае число x разлагается в сумму по степени числа 2:

$$x = \pm(\alpha_n \cdot 2^n + \ldots + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0 + \beta_1 \cdot 2^{-1} + \beta_2 \cdot 2^{-2} + \ldots + \beta_m \cdot 2^{-m} + \ldots).$$

Пример 1.13. Запись $x = 20.5$ в двоичной системе счисления. Для этого разложим его в сумму по степени 2:

$$x = 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1}.$$

Опуская степени числа 2, получаем $x = (10100.1)_2$. Здесь нижний индекс 2 указывает на основание системы счисления.

Для хранения числа в памяти компьютера отводится поле стандартной длины (машинное слово), в котором число записывают в виде последова-

тельности двоичных цифр. По форме представления, способу хранения и реализации арифметических операций существенно различаются два типа используемых в компьютерах чисел: целые и вещественные числа.

1.4.2. *Представление целых чисел*

Целое число n представляют в виде

$$n = \pm(\alpha_L \cdot 2^L + \ldots + \alpha_1 \cdot 2^1 + \alpha_0 \cdot 2^0), \qquad (1.16)$$

где L – некоторое стандартное для компьютера целое число, α_i – двоичные цифры. Всего для хранения числа n отводят $s = L + 2$ разрядов (один из них для хранения знака).

Из представления (1.16) видно, что максимальное по модулю целое число, представимое в компьютере, есть $n_{\max} = 2^L + \ldots + 2^1 + 2^0 = 2^{L+1} - 1$. Обычно оно не очень велико. Например, при стандартном формате записи целых чисел в компьютерах $s = 32$ и $n_{\max} = 2^{31} - 1 \approx 2 \cdot 10^9$.

Операции сложения, вычитания и умножения над целыми числами реализованы так, что если результат не превышает по модулю число n_{max}, то он получается точным. Отметим, однако, следующую неприятную особенность. Если модуль результата превышает $n_{\max}$, то на компьютере эта ситуация может не доводиться до сведения пользователя, происходит присвоение результату некоторого значения (меньшего $n_{\max}$ по модулю) и вычисления продолжаются далее.

1.4.3. *Представление вещественных чисел*

В большинстве современных компьютеров для вещественных чисел принята форма представления с плавающей точкой, когда каждое число представляют в виде

$$x = \pm(\gamma_1 \cdot 2^{-1} + \gamma_2 \cdot 2^{-2} + \ldots + \gamma_l \cdot 2^{-l}) 2^p. \qquad (1.17)$$

Здесь $\gamma_1, \gamma_2, \ldots, \gamma_l$ – двоичные цифры, p – целое число, называемое двоичным порядком.

Число x нормализуется так, чтобы $\gamma_1 = 1$, и поэтому в памяти компьютера хранятся только значащие цифры. Число $\mu = \pm(\gamma_1 \cdot 2^{-1} + \gamma_2 \cdot 2^{-2} + \ldots + \gamma_l \cdot 2^{-l})$ называется мантиссой числа x. Количество l цифр, которое отводится для записи мантиссы, называемое разрядностью мантиссы, зависит от конструктивных особенностей конкретной вычислительной машины, но всегда является конечным. Порядок также записывают как двоичное целое число $p = \pm(\sigma_l \sigma_{l-1} \ldots \sigma_0)_2$, для хранения которого в машинном слове отводится $l + 2$ двоичных разрядов.

Поскольку нуль – не нормализуемое число (его нельзя представить в виде (1.17) при $\gamma_1 \neq 0$), его для хранения записывают особым способом.

Пример 1.14. Представим число $x = 20.5$ в двоичной системе счисления в нормализованной форме с плавающей точкой. Так как $x = (10100.1)_2$ (см. пример 1.13), то, перемещая двоичную точку на пять позиций влево, получаем $x = (0.101001)_2 \cdot 2^5$.

На основании имеющихся сведений о представлении чисел в компьютере можно сделать ряд важных выводов.

1. В компьютере представимы не все числа, а лишь конечный набор рациональных чисел специального вида. Эти числа образуют представимое множество компьютера. Для всех остальных чисел x возможно лишь их приближенное представление с ошибкой, которую принято называть ошибкой представления (или ошибкой округления). Обычно приближенное представление числа x в компьютере обозначают как $x^* = \mathrm{fl}(x)$[1]. Если округление производят по дополнению, то граница относительной погрешности представления равна единице первого отброшенного разряда мантиссы, т.е. $\delta'(x^*) = \varepsilon_M = 2^{-l}$ (порядок числа влияет на относительную погрешность представления). Если же округление производят усечением, то $\delta'(x^*) = \varepsilon_M = 2^{1-l}$. Величина ε_M играет в вычислениях на компьютере фундаментальную роль; ее называют относительной точностью компьютера, а также машинной точностью (или машинным эпсилон). Всюду в дальнейшем ε_M – это относительная точность компьютера. Заметим, что значение этой величины определяется разрядностью мантиссы и способом округления.

Важно с самого начала иметь четкое представление о том, что почти наверняка в представимом множестве чисел компьютера нет числа y, являющегося решением поставленной задачи. Лучшее, что можно попытаться сделать, – это найти его представление $y^* = \mathrm{fl}(y)$ с относительной точностью порядка ε_M.

Полезно отметить, что среди представленных в компьютере чисел нет не только ни одного иррационального (в том числе и таких важных постоянных, как π, e, $\sqrt{2}$), но и даже такого широко используемого в вычислениях числа, как 0.1. Дело в том, что двоичная запись числа 0.1 является бесконечной периодической дробью: $0.1 = (0.0001100110011\ldots)_2$. Поэтому это число всегда представляется в компьютере приближенно, с погрешностью, вызванной необходимостью округления.

2. Диапазон изменения чисел в компьютере ограничен. В самом деле, так как $\gamma_1 = 1$, то из (1.17) следует, что для мантиссы μ справедливы

[1] fl – начальные буквы английского слова floating – «плавающий».

оценки $0.5 \le |\mu| < 1$. В то же время для представления в компьютере порядка p используется конечное число $(l + 1)$ двоичных цифр и поэтому $|p| \le p_{\max} = 2^{l+1} - 1$. Таким образом, для всех представимых в компьютере чисел x (за исключением нуля) имеем

$$0 < X_0 \le |x| < X_\infty,$$

где $X_0 = 2^{-(p_{\max}+1)}$, $X_\infty = 2^{p_{\max}}$. Заметим, что диапазон представления чисел в компьютере всецело определяется разрядностью порядка.

3. Все числа x, по модулю большие X_∞, не представимы в компьютере и могут рассматриваться как машинная бесконечность. Попытка получить такое число приводит к аварийному останову компьютера по переполнению. Все числа x, по модулю меньшие X_0, для вычислительной машины не различимы и представляются как нуль (машинный нуль). Получение числа x такого, что $|x| < X_0$, называют исчезновением порядка (или антипереполнением). Обычно при исчезновении порядка автоматически полагается $\mathrm{fl}(x) = 0$ и вычисления продолжаются.

Не следует смешивать машинную точность ε_M с минимальным положительным представимым в компьютере числом $X_0 << \varepsilon_M$.

4. На машинной числовой оси (см. рис. 1.1) числа расположены неравномерно. Плотность их возрастает по мере приближения к нулю и падает с удалением от нуля. Чтобы убедиться в этом, заметим, что расстояние от одного представимого на компьютере числа x до другого ближайшего представимого числа равно единице последнего разряда мантиссы, умноженной на 2^p, т.е. равно 2^{p-t}. Так как t фиксировано, то расстояние уменьшается с уменьшением порядка p и возрастает с увеличением p.

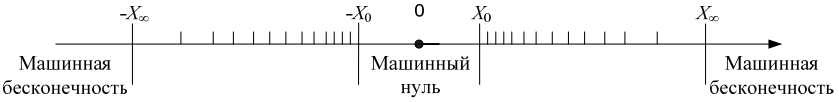


Рис. 1.1

Согласно принятому в настоящее время стандарту IEEE[1] вещественные числа (при вычислениях с одинаковой точностью) представляются с $t = 23$ разрядами мантиссы и с $s = 7$ разрядами показателя. Представимые числа расположены в диапазоне $10^{-38} \le |x| \le 3 \cdot 10^{38}$, вполне достаточном для большинства приложений. Однако разрядность мантиссы невелика

[1] Стандарт IEEE (IEEE Floating Point Standard) был принят в 1985 г. Он точно определяет, как выполняется операция округления, как следует поступать при наличии переполнения, исчезновении порядка, извлечении корня из отрицательного числа и т.д.

($t = 23$) и поэтому $\varepsilon_M \approx 10^{-7}$; в десятичной арифметике это эквивалентно тому, что мантисса содержит семь десятичных цифр. Правда, в компьютерах, поддерживающих стандарт IEEE, сами арифметические операции над числами реализованы со значительно большим числом разрядов и округление до 23 разрядов мантиссы происходит при записи результата операции в память компьютера.

1.4.4. *Арифметические операции над числами с плавающей точкой*

Правила выполнения арифметических операций в двоичной системе счисления чрезвычайно просты и легко реализуются на компьютере. Однако в силу ограниченной разрядности мантиссы операции сложения, вычитания, умножения и деления над представимыми в компьютере вещественными числами не могут быть реализованы точно. Дело в том, что арифметические операции над числами, мантиссы которых содержат t разрядов, приводят, как правило, к результатам, содержащим более t разрядов. Округление результата до t разрядов и служит главным источником погрешности. Для того чтобы отличать машинные арифметические операции от идеальных математических операций +, –, ×, :, будем обозначать их через (+), (–), (×), (:). Игнорируя несущественные детали, можно считать, что результат машинной арифметической операции совпадает с результатом точного выполнения той же операции с погрешностью, приближенно равной погрешности округления. Таким образом,

$$\Delta'(a\ (+)\ b) \approx |a + b|\varepsilon_M,\ \Delta'(a\ (-)\ b) \approx |a - b|\varepsilon_M,$$
$$\Delta'(a\ (\times)\ b) \approx |a \times b|\varepsilon_M,\ \Delta'(a\ (:)\ b) \approx |a : b|\varepsilon_M.$$

Конечно, в некоторых ситуациях округление может отсутствовать. Например, полезно знать, что умножение и деление числа на целую степень числа 2 выполняется на компьютере точно, так как в этом случае мантисса не меняется.

Пример 1.15. Рассмотрим гипотетический компьютер, в котором числа представляются всего лишь с шестью двоичными разрядами мантиссы, а округление производится по дополнению. Пусть на таком компьютере вычисляются сумма и произведение двух представимых на нем чисел $a = 20.5 = (10100.1)_2$ и $b = 1.75 = (1.11)_2$. Производим вычисления в двоичной арифметике:

$$a + b = (10100.1)_2 + (1.11)_2 = (10110.01)_2,$$
$$a \times b = (10100.1)_2 \times (1.11)_2 = (1000011.111)_2.$$

После округления до шести значащих цифр получим $a\,(+)\,b = (10110.1)_2 = 22.5$, $a\,(\times)\,b = (100100.)_2 = 36$. Очевидно, что эти результаты отличаются от точных значений $a + b = 22.25$, $a \times b = 35.875$.

Заметим, что машинные арифметические операции обладают иными свойствами, нежели обычные математические операции. Например, не выполняется известное правило арифметики «от перемены мест слагаемых сумма не меняется». Покажем это на примере.

Пример 1.16. Пусть вычисления производятся на компьютере из примера 1.15, причем $a = (1.)_2$, $b = c = (0.000001)_2$. Тогда $a + b = (1.000001)_2$ и после округления имеем $a\,(+)\,b = (1.00001)_2$. Далее, $[a\,(+)\,b] + c = (1.000011)_2$ и после округления получим $[a\,(+)\,b]\,(+)\,c = (1.00010)_2$. Сложение в ином порядке дает $c\,(+)\,b = (0.000010)_2$, $[c\,(+)\,b] + a = (1.00001)_2$. Таким образом, $[a\,(+)\,b] + c \neq [c\,(+)\,b] + a$.

1.4.5. *Удвоенная точность*

На большинстве компьютеров возможна реализация арифметических действий над числами, разрядность мантисс которых примерно вдвое превосходит стандартную разрядность t. Это приводит к существенному повышению машинной точности. Например, на компьютерах, удовлетворяющих стандарту IEEE, в режиме удвоенной точности $t = 52$, $s = 11$ и $\varepsilon_M \approx 10^{-16}$. Для сравнения напомним, что обычная точность этих компьютеров есть $\varepsilon_M \approx 10^{-7}$, так что следует говорить не об удвоении точности, а о повышении точности на много порядков.

На тех компьютерах, где арифметика реализована аппаратурно, время исполнения программ возрастает несущественно (часто коэффициент увеличения близок к единице). Если же реализация вычислений с удвоенной мантиссой осуществляется программным образом, то время счета увеличивается в несколько раз. Поскольку для хранения числа с удвоенной мантиссой отводится два машинных слова, в этом случае вдвое возрастает используемая память.

Есть также режим увеличенной точности («extended precision»), для которого $t = 63$ и $s = 14$. В этом режиме представимые числа находятся в диапазоне $10^{-4964} \le |x| \le 10^{4964}$ и $\varepsilon_M \sim 10^{-19}$. Подчеркнем, что использование режимов удвоенной или увеличенной точности отнюдь не позволяет ликвидировать погрешности округления, а только лишь уменьшает их влияние.

1.4.6. *Вычисление машинного эпсилон*

Для приближенного вычисления величины ε_M удобно воспользоваться следующим определением. Машинное эпсилон – это минимальное из представимых на компьютере положительных чисел ε, для которых $1\ (+)\ \varepsilon > 1$.

Величину ε_M можно оценить непосредственно в ходе вычислительного процесса. Для этого достаточно включить в программу фрагмент, реализующий следующий метод. Полагая $\varepsilon^{(0)} = 1$, следует вычислять последовательно $\varepsilon^{(1)} = 0.5\varepsilon^{(0)}$, $\varepsilon^{(2)} = 0.5\varepsilon^{(1)}$, …, $\varepsilon^{(n)} = 0.5\varepsilon^{(n-1)}$, …, запоминая получающиеся значения $\varepsilon^{(n)}$ в памяти компьютера и проверяя каждый раз выполнение неравенства $1\ (+)\ \varepsilon^{(n)} > 1$. Как только при некотором n окажется, что $1\ (+)\ \varepsilon^{(n)} = 1$, следует положить $\varepsilon_M = \varepsilon^{(n-1)}$ и перейти к следующему этапу вычислений. Хотя полученное таким способом значение может отличаться от ε_M в два раза, обычно оно используется так, что эта погрешность не имеет значения.

Пример 1.17. Покажем, что $\varepsilon_M = 2^{-6} = (0.000001)_2$ – машинное эпсилон для компьютера из примера 1.15. В самом деле, $1 + \varepsilon_M = (1.000001)_2$, и после округления имеем $1\ (+)\ \varepsilon_M = (1.00001)_2 > 1$. Если же к 1 добавить любое положительное $\varepsilon < \varepsilon_M$, то в седьмом разряде результата будет стоять нуль, и после округления получим $1\ (+)\ \varepsilon = 1$.

2. ПРЯМЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

2.1. Постановка задачи

В вычислительной линейной алгебре выделяют 4 основные задачи:

1) решение СЛАУ;

2) вычисление определителей;

3) нахождение обратных матриц;

4) определение собственных значений и собственных векторов.

Задачу 1 в линейной алгебре называют первой основной задачей [2]. Эффективность способа решения системы $\mathbf{Ax} = \mathbf{b}$ во многом зависит от структуры матрицы $\mathbf{A}$: размера, обусловленности, симметричности, заполненности (т.е. соотношения между числом ненулевых и нулевых элементов), специфики расположения ненулевых элементов в матрице и др.

При рассмотрении будем полагать, что матрица $\mathbf{A}$ задана и является невырожденной. Известно, что в этом случае решение системы существует, единственно и устойчиво по входным данным. Это означает, что рассматриваемая задача корректна.

Все методы решения линейных алгебраических систем можно разбить на два класса: прямые и итерационные. Прямыми называются методы, которые приводят к решению за конечное число арифметических операций. Если операции реализуются точно, то и решение также будет точным (в связи с чем к классу прямых методов применяют еще название точные методы). Итерационными методами являются методы, в которых точное решение может быть получено лишь в результате бесконечного повторения, как правило, единообразных действий.

Данная глава посвящена прямым методам решения системы линейных алгебраических уравнений. Итерационные методы решения СЛАУ будут рассмотрены в следующей главе.

Уделим основное внимание задаче вычисления вектора $\mathbf{x}$, являющегося решением системы $\mathbf{Ax} = \mathbf{b}$. Будем предполагать, что матрица $\mathbf{A}$ задана и является невырожденной ($\det\mathbf{A} \neq 0$). Известно, что в этом случае решение системы существует, единственно и устойчиво по входным данным. Это означает, что рассматриваемая задача корректна.

Пусть $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_N^*)^T$ – приближенное решение СЛАУ (T – символ транспонирования). В этой и следующих главах будем стремиться к получению решения, для которого погрешность $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ мала (количественные характеристики «величины» погрешности будут введены в следующем параграфе). Тем не менее, заметим, что качество полученного

решения далеко не всегда характеризуется тем, насколько мала погрешность $\mathbf{x} - \mathbf{x}^*$. Иногда вполне удовлетворительным является критерий малости невязки $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^*$. Вектор $\mathbf{r}$ показывает, насколько отличается правая часть системы от левой, если подставить в нее приближенное решение. Заметим, что $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}^* = \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$ и поэтому погрешность и невязка связаны равенством

$$\mathbf{e} = \mathbf{x} - \mathbf{x}^* = \mathbf{A}^{-1}\mathbf{r}. \qquad (2.1)$$

2.2. Норма вектора

Решением СЛАУ является вектор $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$, который будем рассматривать как элемент векторного пространства $\boldsymbol{C}^N$. Приближенное решение $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_N^*)^T$ и погрешность $\mathbf{e} = \mathbf{x} - \mathbf{x}^* = (x_1 - x_1^*, x_2 - x_2^*, \ldots, x_N - x_N^*)^T$ также являются элементами пространства $\boldsymbol{C}^N$. Для того чтобы анализировать методы решения СЛАУ, необходимо уметь количественно оценивать «величины» векторов $\mathbf{x}^*$ и $\mathbf{x} - \mathbf{x}^*$, а также векторов $\mathbf{b}$ и $\mathbf{b} - \mathbf{b}^*$, где $\mathbf{b}^* = (b_1^*, b_2^*, \ldots, b_N^*)^T$ – вектор приближенно заданных правых частей. Удобной для этой цели количественной характеристикой является широко используемое понятие нормы вектора.

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

1) $\|\mathbf{x}\| \geq 0$, причем $\|\mathbf{x}\| = 0$ тогда и только тогда, когда $\mathbf{x} = 0$;

2) $\|\alpha\mathbf{x}\| = |\alpha| \, \|\mathbf{x}\|$ для любого вектора $\mathbf{x}$ и любого числа α;

3) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ для любых векторов $\mathbf{x}$ и $\mathbf{y}$.

Последнее неравенство принято называть неравенством треугольника.

Заметим, что таким 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:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{N} |x_i| \, ;$$

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{N} |x_i|^2\right)^{1/2} \text{ – евклидова норма}^1;$$

[1] Эта норма является естественным обобщением на случай N-мерного пространства понятия длины вектора в двух- и трехмерных геометрических пространствах. Поэтому ее называют евклидовой.

$$\| \mathbf{x} \|_\infty = \max_{1 \le i \le N} | x_i | .$$

Первые две из них являются частными случаями более общей нормы:

$$\| \mathbf{x} \|_p = (\sum_{i=1}^{N} | x_i |^p)^{1/p}, \; p \ge 1 \qquad (2.2)$$

(при $p = 1$ и $p = 2$), а последняя, как можно показать, получается из нормы (2.2) предельным переходом при $p \to \infty$.

Также справедливы неравенства

$$\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1 \le m\|\mathbf{x}\|_\infty, \qquad (2.3)$$

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 2.1. Найдем $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, $\|\mathbf{x}\|_\infty$ для вектора $x = (0.12, -0.15, 0.16)^T$.

По приведенным выше формулам определяем

$\|\mathbf{x}\|_1 = 0.12 + 0.15 + 0.16 = 0.43$,

$\|\mathbf{x}\|_2 = (0.12^2 + 0.15^2 + 0.16^2)^{1/2} = 0.25$,

$\|\mathbf{x}\|_\infty = \max\{0.12, 0.15, 0.16\} = 0.16$.

2.3. Скалярное произведение

Скалярным произведением векторов $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ и $\mathbf{y} = (y_1, y_2, \ldots, y_N)^T$ называется величина

$$(\mathbf{x}, \mathbf{y}) = x_1 y_1 + x_2 y_2 + \ldots + x_N y_N = \sum_{i=1}^{N} x_i y_i .$$

Нетрудно заметить, что $\|\mathbf{x}\|_2 = (\mathbf{x}, \mathbf{x})^{1/2}$. Когда векторы $\mathbf{x}$ и $\mathbf{y}$ имеют комплексные компоненты, скалярное произведение понимают так:

$$(\mathbf{x}, \mathbf{y}) = x_1 \overline{y}_1 + x_2 \overline{y}_2 + \ldots + x_N \overline{y}_N = \sum_{i=1}^{N} x_i \overline{y}_i ,$$

где черта означает комплексное сопряжение.

2.4. Абсолютная и относительная погрешность вектора

Далее будем всюду считать, что в пространстве N-мерных векторов $\mathbf{C}^N$ введена и фиксирована некоторая норма $\|\mathbf{x}\|$ (например, одна из норм $\|\mathbf{x}\|_p$, $1 \le p \le \infty$). В этом случае в качестве меры степени близости векторов $\mathbf{x}$ и $\mathbf{x}^*$ естественно использовать величину $\|\mathbf{x} - \mathbf{x}^*\|$, являющуюся аналогом расстояния между точками $\mathbf{x}$ и $\mathbf{x}^*$. Введем абсолютную и относительную погрешности вектора $\mathbf{x}^*$ с помощью формул

$$\Delta(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\|,$$

$$\delta(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\| / \|\mathbf{x}\|.$$

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

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

2.5. Сходимость по норме

Пусть $\mathbf{x}^{(m)}$ ($m = 1, 2, \ldots, \infty$) – последовательность векторов $\mathbf{x}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \ldots, x_N^{(m)})^T$. Говорят, что последовательность векторов $\mathbf{x}^{(m)}$ сходится к вектору $\mathbf{x}$ при $m \to \infty$ ($\mathbf{x}^{(m)} \to \mathbf{x}$ при $m \to \infty$), если $\Delta(\mathbf{x}^{(m)}) = \|\mathbf{x}^{(m)} - \mathbf{x}\| \to 0$ при $m \to \infty$.

Сам факт наличия или отсутствия сходимости $\mathbf{x}^{(m)}$ к $\mathbf{x}$ при $m \to \infty$ 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. Например, для норм $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$, $\|\mathbf{x}\|_\infty$ это вытекает из неравенства (2.3). Более того, $\mathbf{x}^{(m)} \to \mathbf{x}$ при $m \to \infty$ тогда и только тогда, когда для всех $i = 1, 2, \ldots, N$ имеем $\mathbf{x}_i^{(m)} \to \mathbf{x}_i$ при $m \to \infty$, т.е. сходимость по норме в $\boldsymbol{C}^N$ эквивалентна покомпонентной (покоординатной) сходимости.

2.6. Норма матрицы

Величина

$$\| \mathbf{A} \| = \max_{\mathbf{x} \neq 0} \frac{\| \mathbf{A}\mathbf{x} \|}{\| \mathbf{x} \|} \tag{2.4}$$

называется нормой матицы $\mathbf{A}$, подчиненной норме векторов, в $\boldsymbol{C}^N$.

Заметим, что множество всех квадратных матриц размера $N \times N$ является векторным пространством. Можно показать, что введенная в этом пространстве формулой (2.4) норма обладает следующими свойствами, аналогичными свойствам нормы вектора:

1) $\|\mathbf{A}\| \geq 0$, причем $\|\mathbf{A}\| = 0$ тогда и только тогда, когда $\mathbf{A} = 0$;

2) $\|\alpha\mathbf{A}\| = |\alpha| \, \|\mathbf{A}\|$ для любой матрицы $\mathbf{A}$ и любого числа α;

3) $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ для любых матриц $\mathbf{A}$ и $\mathbf{B}$.

Дополнительно к этому верны следующие свойства:

4) $\|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$ для любых матриц $\mathbf{A}$ и $\mathbf{B}$;

5) для любой матрицы $\mathbf{A}$ и любого вектора $\mathbf{x}$ справедливо неравенство

$$\|\mathbf{A}\,\mathbf{x}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{x}\|. \tag{2.5}$$

Как следует из определения (2.4), каждой из векторных норм $\|\mathbf{x}\|$ соответствует своя подчиненная норма матрицы $\mathbf{A}$. Известно, в частности, что

нормам $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$ и $\|\mathbf{x}\|_\infty$ подчинены нормы $\|\mathbf{A}\|_1$, $\|\mathbf{A}\|_2$ и $\|\mathbf{A}\|_\infty$, вычисляемые по формулам

$$\| \mathbf{A} \|_1 = \max_{1 \le j \le N} \sum_{i=1}^{N} | a_{ij} | , \tag{2.6}$$

$$\| \mathbf{A} \|_2 = \max_{1 \le j \le N} \sqrt{\lambda_j (\mathbf{A}^T \mathbf{A})} , \tag{2.7}$$

где $\lambda_j(\mathbf{A}^T\mathbf{A})$ – собственные числа[1] матрицы $\mathbf{A}^T\mathbf{A}$;

$$\| \mathbf{A} \|_\infty = \max_{1 \le i \le N} \sum_{j=1}^{N} | a_{ij} | \tag{2.8}$$

Нормы $\|\mathbf{A}\|_1$ и $\|\mathbf{A}\|_\infty$ вычисляются просто. Для вычислнения $\|\mathbf{A}\|_1$ нужно найти сумму модулей элементов каждого из столбцов матрицы **A**, а затем выбрать максимальную из этих сумм. Для получения значения $\|\mathbf{A}\|_\infty$ нужно аналогичным образом поступить со строками матрицы **A**.

Как правило, вычислить значение нормы $\|\mathbf{A}\|_2$ бывает трудно, так как для этого следует искать собственные числа λ_j. Для оценки величины $\|\mathbf{A}\|_2$ можно, например, использовать неравенство

$$\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_E. \tag{2.9}$$

Здесь $\| \mathbf{A} \|_E = \left(\sum_{i,j=1}^{N} | a_{ij} |^2 \right)^{1/2}$ – величина, называемая евклидовой нормой матрицы **A**. Эту норму также называют нормой Фробениуса, шуровской нормой, а также нормой Э. Шмидта [3].

Норма (2.4) имеет простую геометрическую интерпретацию. Для того чтобы её привести, заметим, что операцию умножения матрицы **A** на вектор **x** можно рассматривать как преобразование, которое переводит вектор **x** в новый вектор $\mathbf{y} = \mathbf{A}\mathbf{x}$. Если значение $\|\mathbf{x}\|$ интерпретируется как длина вектора **x**, то величина $\|\mathbf{A}\mathbf{x}\| / \|\mathbf{x}\|$ есть коэффициент растяжения вектора **x** под действием матрицы **A**. Таким образом, величина

$$k_{\max} = \| \mathbf{A} \| = \max_{\mathbf{x} \ne 0} \frac{\| \mathbf{A}\mathbf{x} \|}{\| \mathbf{x} \|} \tag{2.10}$$

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

$$k_{\min} = \| \mathbf{A}^{-1} \|^{-1} = \min_{\mathbf{x} \ne 0} \frac{\| \mathbf{A}\mathbf{x} \|}{\| \mathbf{x} \|} . \tag{2.11}$$

---

[1] Число λ называется собственным числом матрицы **A**, если $\det(\mathbf{A} - \lambda\mathbf{E}) = 0$ (**E** – единичная матрица). Каждая матрица порядка *N* имеет ровно *N* собственных чисел (вообще говоря, комплексных) с учетом их кратности.

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

Пример 2.2. Для матрицы

$$\mathbf{A} = \begin{pmatrix} 0.1 & -0.4 & 0 \\ 0.2 & 0 & -0.3 \\ 0 & 0.1 & 0.3 \end{pmatrix}$$

найдем $\|\mathbf{A}\|_1$, $\|\mathbf{A}\|_\infty$ и оценим $\|\mathbf{A}\|_2$.

В соответствии с формулами (2.6), (2.8) и неравенством (2.9) имеем

$\|\mathbf{A}\|_1 = \max\{0.1 + 0.2 + 0;\ 0.4 + 0 + 0.1;\ 0 + 0.3 + 0.3\} = 0.6;$

$\|\mathbf{A}\|_\infty = \max\{0.1 + 0.4 + 0;\ 0.2 + 0 + 0.3;\ 0 + 0.1 + 0.3\} = 0.5;$

$\|\mathbf{A}\|_2 \le \|\mathbf{A}\|_E = \left(\sum_{i,j=1}^{3} a_{ij}^2\right)^{1/2} = \sqrt{0.4} \approx 0.63.$

2.7. Обусловленность задачи решения системы линейных алгебраических уравнений

Практика вычислений, проведенных на различных задачах, показала, что решения различных систем линейных алгебраических уравнений обладают разной чувствительностью к погрешностям входных данных. Так же как и другие задачи, задача вычисления решения **x** системы линейных алгебраических уравнений $\mathbf{Ax} = \mathbf{b}$ может быть как хорошо, так и плохо обусловленной.

Исследование обусловленности задачи начнем со случая, когда элементы матрицы **A** считаются заданными точно, а вектор-столбец правой части – приближенно.

Для погрешности приближенного решения системы справедлива оценка

$$\Delta(\mathbf{x}^*) \le \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}\|, \tag{2.12}$$

где $\mathbf{r} = \mathbf{b} - \mathbf{Ax}^*$ – невязка, отвечающая $\mathbf{x}^*$. Для доказательства достаточно взять норму левой и правой частей равенства (2.1) и воспользоваться свойством (2.5).

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

$$\Delta(\mathbf{x}^*) \le \nu_\Delta \Delta(\mathbf{b}^*), \tag{2.13}$$

$$\delta(\mathbf{x}^*) \le \nu_\delta \delta(\mathbf{b}^*), \tag{2.14}$$

где $\nu_\Delta = \|\mathbf{A}^{-1}\|$, $\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{r}\| / \|\mathbf{x}\| = \|\mathbf{A}^{-1}\| \cdot \|\mathbf{Ax}\| / \|\mathbf{x}\|$.

В рассматриваемом случае $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^* = \mathbf{b} - \mathbf{b}^*$ и неравенство (2.12) принимает вид (2.13). Разделив теперь обе части неравенства (2.13) на $\|\mathbf{x}\|$ и записав его в виде

$$\Delta(\mathbf{x}^*) / \|\mathbf{x}\| \le (\|\mathbf{A}^{-1}\|\cdot\|\mathbf{b}\| / \|\mathbf{x}\|)\cdot(\Delta(\mathbf{x}^*) / \|\mathbf{b}\|),$$

приходим к оценке (2.14).

Сделаем несколько замечаний относительно числа обусловленности системы линейных алгебраических уравнений.

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

2. Величина $\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\|\cdot\|\mathbf{r}\| / \|\mathbf{x}\|$ называется естественным числом обусловленности. Она зависит от конкретного решения $\mathbf{x}$ и характеризует коэффициент возможного возрастания относительной погрешности этого решения, вызванного погрешностью задания правой части. Это означает, что $\nu_\delta(\mathbf{x})$ для задачи вычисления решения системы $\mathbf{A}\mathbf{x} = \mathbf{b}$ играет роль относительного числа обусловленности.

3. Полученные оценки (2.13) и (2.14) точны в том смысле, что для системы $\mathbf{A}\mathbf{x} = \mathbf{b}$ с произвольной невырожденной правой частью $\mathbf{b} \ne 0$ найдется сколь угодно близкий к $\mathbf{b}$ приближенно заданный вектор $\mathbf{b}^* \ne \mathbf{b}$, для которого неравенства (2.13) и (2.14) превращаются в равенства.

Вычислим максимальное значение естественного числа обусловленности, используя определение нормы матрицы:

$$\max_{\mathbf{x}\ne 0} \nu_\delta(\mathbf{x}) = \frac{\| \mathbf{A}^{-1} \| \cdot \| \mathbf{A}\mathbf{x} \|}{\| \mathbf{x} \|} = \| \mathbf{A}^{-1} \| \cdot \| \mathbf{A} \| . \qquad (2.15)$$

Полученную величину принято называть стандартным числом обусловленности (или просто числом обусловленности) матрицы $\mathbf{A}$ и обозначать через $\nu(\mathbf{A})$ или $\mathrm{cond}(\mathbf{A})$. Таким образом,

$$\nu(\mathbf{A}) = \mathrm{cond}(\mathbf{A}) = \|\mathbf{A}^{-1}\|\cdot\|\mathbf{A}\|. \qquad (2.16)$$

Следствием оценок (2.13) и (2.14) является оценка

$$\delta(\mathbf{x}^*) \le \mathrm{cond}(\mathbf{A})\cdot\delta(\mathbf{b}^*). \qquad (2.17)$$

Для ее доказательства достаточно воспользоваться оценкой (2.14) и заметить, что в силу равенства (2.15) верно неравенство $\nu_\delta \le \mathrm{cond}(\mathbf{A})$.

Оценка (2.17) точна в том смысле, что для системы $\mathbf{A}\mathbf{x} = \mathbf{b}$ с произвольной невырожденной матрицей $\mathbf{A}$ найдутся правая часть $\mathbf{b} \ne 0$ (и отвечающее этой правой части решение $\mathbf{x}$) и сколь угодно близкий к $\mathbf{b}$ приближенно заданный вектор $\mathbf{b}^* \ne \mathbf{b}$ такие, что неравенство (2.17) превращается в равенство.

Величина $\mathrm{cond}(\mathbf{A})$ является широко используемой количественной мерой обусловленности системы $\mathbf{A}\mathbf{x} = \mathbf{b}$. В частности, систему и матрицу $\mathbf{A}$ принято называть плохо обусловленными, если $\mathrm{cond}(\mathbf{A}) \gg 1$. В силу

последнего замечания и оценки (2.17) для такой системы существуют решения, обладающие чрезвычайно высокой чувствительностью к малым погрешностям задания входного данного **b**. Тем не менее, заметим, что для всякого решения **x** коэффициент ν_δ (**x**) роста относительной погрешности достигает значений, близких к максимально возможному значению cond(**A**).

Отметим следующие, часто используемые, свойства числа обусловленности.

1. Для единичной матрицы cond(**E**) = 1. (Пользуясь тем, что $\mathbf{E}^{-1} = \mathbf{E}$ и $||\mathbf{E}|| = 1$, получаем $\text{cond}(\mathbf{E}) = ||\mathbf{E}^{-1}||\cdot||\mathbf{E}|| = 1$.)

2. Справедливо неравенство $\text{cond}(\mathbf{A}) \geq 1$. (Из равенства $\mathbf{E} = \mathbf{A}\cdot\mathbf{A}^{-1}$, свойства 4 норм матриц и равенства $||\mathbf{E}|| = 1$ следует, что $1 = ||\mathbf{E}|| \leq ||\mathbf{A}^{-1}||\cdot||\mathbf{A}|| = \text{cond}(\mathbf{A})$).

3. Число обусловленности матрицы **A** не меняется при умножении матрицы на произвольное число $\alpha \neq 0$. (Заметим, что $(\alpha\mathbf{A})^{-1} = \alpha^{-1}\mathbf{A}^{-1}$). Поэтому $\text{cond}(\alpha\mathbf{A}) = ||\alpha\mathbf{A}||\cdot||(\alpha\mathbf{A})^{-1}|| = |\alpha|\cdot||\mathbf{A}||\cdot|\alpha|^{-1}\cdot||\mathbf{A}^{-1}|| = \text{cond}(\mathbf{A})$.

Пользуясь приведенной выше геометрической интерпретацией норм матриц **A** и $\mathbf{A}^{-1}$ (см. формулы (2.10) и (2.11)), число обусловленности можно интерпретировать как отношение максимального коэффициента растяжения векторов под действием матрицы **A** к минимальному коэффициенту: $\text{cond}(\mathbf{A}) = k_{\max} / k_{\min}$.

Величина cond(**A**) зависит, вообще говоря, от выбора нормы векторов в пространстве $\boldsymbol{C}^N$. Фактически это есть зависимость максимального коэффициента роста погрешности от способа измерения величины входных данных и решения. В частности, выбору нормы $||\mathbf{x}||_p$ $(1 \leq p \leq \infty)$ отвечает $\text{cond}_p(\mathbf{A}) = ||\mathbf{A}^{-1}||_p\cdot||\mathbf{A}||_p$.

Пример 2.3. Вычислим $\text{cond}_\infty(\mathbf{A})$ для матрицы

$$\mathbf{A} = \begin{pmatrix} 1.03 & 0.991 \\ 0.991 & 0.943 \end{pmatrix}.$$

С начала найдем обратную матрицу

$$\mathbf{A}^{-1} \approx \begin{pmatrix} -87.4 & 91.8 \\ 91.8 & -95.4 \end{pmatrix}.$$

Тогда $\text{cond}_\infty(\mathbf{A}) = ||\mathbf{A}||_\infty\cdot||\mathbf{A}^{-1}||_\infty \approx 2.021\cdot187.2 \approx 378$. Если входные данные для системы уравнений с матрицей **A** содержат относительную погрешность порядка 0.1–1%, то систему можно расценить как плохо обусловленную.

Пример 2.4. Рассмотрим систему уравнений

$$1.03x_1 + 0.991x_2 = 2.51,$$
$$0.991x_1 + 0.943x_2 = 2.41$$

с матрицей из примера 2.3. Ее решением является $x_1 = 1.981$, $x_2 = 0.4735$. Правая часть системы известна с точностью до 0.005, если считать, что числа 2.51 и 2.41 получены округлением «истинных» значений при вводе в память трехзначного десятичного компьютера. Как влияет погрешность во входных данных такого уровня на погрешность решения? Возмутим каждую из компонент вектора $\mathbf{b} = (2.51, 2.41)^T$ на 0.005, взяв $\mathbf{b}^* = (2.505, 2.415)^T$. Решением системы, отвечающим $\mathbf{b}^*$, является теперь $x_1^* = 2.877$, $x_2^* = -0.4629$. Таким образом, решение оказалось полностью искаженным. Относительная погрешность правой части $\delta(\mathbf{b}^*) = \|\mathbf{b} - \mathbf{b}^*\|_\infty / \|\mathbf{b}\|_\infty = 0.005 / 2.51 \approx 0.2\%$ привела к относительной погрешности решения $\delta(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\|_\infty / \|\mathbf{x}\|_\infty \approx 0.9364 / 1.981 \approx 47.3\%$. 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 $x_1 = 1.981$, $x_2 = 0.4735$ и получим

$$\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\|_\infty \|\mathbf{b}\|_\infty / \|\mathbf{x}\|_\infty \approx 187.2 \cdot 2.51 / 1.981 \approx 237.$$

Таким образом, на поставленный вопрос следует ответить отрицательно.

Можно дать геометрическую интерпретацию рассмотренного примера. Каждому уравнению 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 (см. рис. 2.1).

Пример 2.5. Традиционным примером очень плохо обусловленной матрицы является матрица Гильберта[1] – матрица $\mathbf{H}$ с элементами $h_{ij} = 1/(i + j - 1)$, $i, j = 1, \ldots, N$.

Из табл. 2.1, заимствованной из [4], видно, что для матрицы $\mathbf{H}$ даже сравнительно невысокого порядка число обусловленности оказывается чрезвычайно большим.

[1] Давид Гильберт (1862–1943) – немецкий математик, исследования которого оказали большое влияние на развитие современной математики.

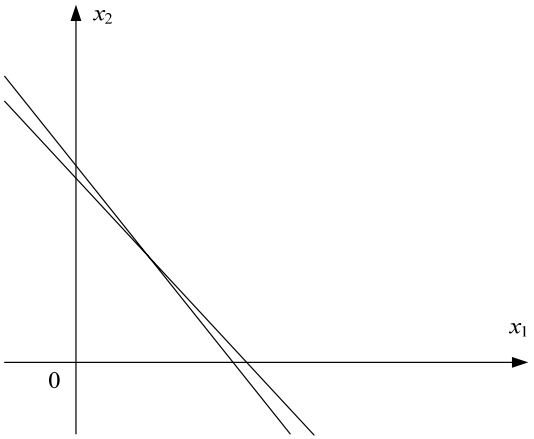


Рис. 2.1

Таблица 2.1

Порядок матрицы Гильберта	2	3	4	5	6	7	8	9	10
Приближенное значение cond(**A**)	$2\cdot10^{1}$	$5\cdot10^{2}$	$2\cdot10^{4}$	$5\cdot10^{5}$	$2\cdot10^{7}$	$5\cdot10^{8}$	$2\cdot10^{10}$	$5\cdot10^{11}$	$2\cdot10^{13}$

В табл. 2.2 приведен вектор решения СЛАУ (с точностью до шести знаков после точки) с матрицей Гильберта методом Гаусса при порядке матрицы 10, 11, 12, 13. (При вычислениях использовалась двойная точность.) Из таблицы видно, что с увеличением порядка, полученное решение все сильнее отклоняется от истинного решения 1, 2 , …, *N*.

Таблица 2.2

Порядок матрицы Гильберта	10	11	12	13
Вектор решения	1.000000 2.000000 3.000000 3.999991 5.000398 5.998902 7.001807 7.001807 9.000921 9.999797	1.000000 1.999994 3.000148 3.998339 5.009928 5.965003 7.076365 7.895707 9.086759 9.959809 11.007948	1.000000 1.999982 3.000552 3.992507 5.054747 5.760160 7.666531 6.769259 10.408343 8.970460 11.427339 11.923121	1.000000 2.000077 2.997022 4.049612 4.554488 8.415663 –1.422679 27.516447 –21.369275 41.366039 –9.619528 19.808824 11.703310

До сих пор мы предполагали, что матрица **A** задана точно. Однако на практике это часто не так. Как выясняется, введенная выше величина cond(**A**) характеризует также и чувствительность решений системы к малым погрешностям задания элементов матрицы **A**. В подтверждение сказанного приведем следующий результат.

Пусть $\mathbf{x}^*$ – точное решение системы $\mathbf{A}^*\mathbf{x}^* = \mathbf{b}$ с приближенно заданной матрицей $\mathbf{A}^*$. Тогда верна оценка относительной погрешности:

$$\delta^*(\mathbf{x}^*) \le \mathrm{cond}(\mathbf{A})\cdot\delta(\mathbf{A}^*), \qquad (2.18)$$

где $\delta^*(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\| / \|\mathbf{x}^*\|$, $\delta(\mathbf{A}^*) = \|\mathbf{A} - \mathbf{A}^*\| / \|\mathbf{A}\|$.

В данном случае невязка **r** имеет вид $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^* = \mathbf{A}^*\mathbf{x}^* - \mathbf{A}\mathbf{x}^* = (\mathbf{A}^* - \mathbf{A})\mathbf{x}^*$. Применяя неравенство (2.12), получаем цепочку неравенств

$$\delta^*(\mathbf{x}^*) = \frac{\|\mathbf{x} - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} \le \frac{\|\mathbf{A}^{-1}\|\cdot\|(\mathbf{A}^* - \mathbf{A})\mathbf{x}^*\|}{\|\mathbf{x}^*\|} \le \frac{\|\mathbf{A}^{-1}\|\cdot\|\mathbf{A}^* - \mathbf{A}\|\cdot\|\mathbf{x}^*\|}{\|\mathbf{x}^*\|} =$$

$$= \mathrm{cond}(\mathbf{A})\cdot\delta(\mathbf{A}^*).$$

Распространенным является представление о том, что по значению определителя матрицы **A** можно судить о степени близости системы уравнений к вырожденной или об обусловленности системы. Для того чтобы убедиться в ошибочности этого мнения, умножим каждое из уравнений СЛАУ $\mathbf{A}\mathbf{x} = \mathbf{b}$ на постоянную $\alpha \ne 0$. Ясно, что такое преобразование никак не меняет решение системы и его чувствительность к малым относительным погрешностям в данных. Однако определитель умножается на число α^N (т.к. $\det(\alpha\mathbf{A}) = \alpha^N\cdot\det\mathbf{A}$) и поэтому с помощью выбора α может быть сделан как угодно большим или малым. Подчеркнем, что при этом число обусловленности cond(**A**) не меняется в силу свойства 3.

Вычисление чисел обусловленности $\nu_\delta(\mathbf{x}) = \|\mathbf{A}^{-1}\|\cdot\|\mathbf{b}\| / \|\mathbf{x}\|$ и $\mathrm{cond}(\mathbf{A}) = \|\mathbf{A}^{-1}\|\cdot\|\mathbf{A}\|$ непосредственно по указанным формулам предполагает предварительное вычисление обратной матрицы $\mathbf{A}^{-1}$. Вследствие большой трудоемкости этой операции ($\sim 2N^3$) на практике избегают такого способа вычисления. При этом важно отметить, что в большинстве случаев достаточно лишь грубой оценки числа обусловленности с точностью до порядка. С эффективными методами, дающими оценки величин $\nu_\delta(\mathbf{x})$ и cond(**A**), можно познакомиться в [5–7].

Проверить чувствительность решения системы $\mathbf{A}\mathbf{x} = \mathbf{b}$ к погрешностям можно и экспериментально. Для этого достаточно решить задачу несколько раз с близкими к **b** правыми частями $\mathbf{b}^{(1)}$, $\mathbf{b}^{(2)}$,…, $\mathbf{b}^{(n)}$. Можно ожидать, что величина $\nu'_\delta = \max\{\delta(\mathbf{x}^{(l)}) / \delta(\mathbf{b}^{(l)})\}$, $1 \le l \le n$ даст оценку значения $\nu_\delta(\mathbf{x})$. Во всяком случае, она даст оценку снизу, так как $\nu'_\delta \le \nu_\delta(\mathbf{x}) \le \mathrm{cond}(\mathbf{A})$.

2.8. Метод Крамера

Один из самых старых методов решения СЛАУ был предложен Крамером[1] (формулы Крамера) в 1750 году. Данный метод применим только в случае систем линейных уравнений, где число переменных совпадает с числом уравнений. Кроме того, необходимо ввести ограничения на коэффициенты системы. Необходимо, чтобы все уравнения были линейно независимы, т.е. ни одно уравнение не являлось бы линейной комбинацией остальных. Для этого необходимо, чтобы определитель матрицы системы не равнялся нулю ($\det\mathbf{A} \neq 0$). Действительно, если какое-либо уравнение системы есть линейная комбинация остальных, то если к элементам какой-либо строки прибавить элементы другой, умноженные на какое-либо число, с помощью линейных преобразований можно получить нулевую строку. Определитель в этом случае будет равен нулю.

Данный способ позволяет находить неизвестные компоненты вектора **x** в виде дробей, знаменателем которых является определитель матрицы системы, а числителем – определители матриц $\mathbf{A}_i$, полученные из матрицы **A** заменой столбца коэффициентов при вычисляемом неизвестном столбцом свободных членов ($x_i = \det\mathbf{A}_i / \det\mathbf{A}$ для $i = 1, 2,\ldots, N$).

Пример 2.6. Решить с помощью формул Крамера СЛАУ

$$\begin{pmatrix} 2 & 1 & 3 \\ 4 & -1 & 8 \\ -6 & 3 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13 \\ 26 \\ 6 \end{pmatrix}.$$

$$\det\mathbf{A} = \begin{vmatrix} 2 & 1 & 3 \\ 4 & -1 & 8 \\ -6 & 3 & 2 \end{vmatrix} = 2\cdot(-1)^{(1+1)}[(-1\cdot 2)-(3\cdot 8)]+$$

$$+1\cdot(-1)^{(1+2)}[(4\cdot 2)-(-6\cdot 8)]+3\cdot(-1)^{(1+3)}[(4\cdot 3)-(-1\cdot -6)]=-90;$$

$$\det\mathbf{A}_1 = \begin{vmatrix} 13 & 1 & 3 \\ 26 & -1 & 8 \\ 6 & 3 & 2 \end{vmatrix} = -90\ ;\ x_1 = \frac{-90}{-90} = 1;$$

[1] Крамер Габриель (1704–1752) – швейцарский математик. Основные труды по высшей алгебре и аналитической геометрии. Был учеником и другом Иоганна Бернулли. Заложил основы теории определителей. Ему принадлежат также исследования по теории алгебраических кривых высших порядков (исследование особых точек, ветвей и т.п.).

$$\det\mathbf{A}_2=\begin{vmatrix}2 & 13 & 3\\ 4 & 26 & 8\\ -6 & 6 & 2\end{vmatrix}=-180;\ x_2=\frac{-180}{-90}=2;$$

$$\det\mathbf{A}_3=\begin{vmatrix}2 & 1 & 13\\ 4 & -1 & 26\\ -6 & 3 & 6\end{vmatrix}=-270;\ x_3=\frac{-270}{-90}=3.$$

Если при реализации этих формул определители вычисляются понижением порядка на основе разложения по элементам какой-нибудь строки или столбца матрицы, то на вычисление определителя N-го порядка будет затрачиваться $N!$ операций умножения. Факториальный рост количества арифметических операций (и вообще говоря, очень быстрый рост) с увеличением размерности задачи называется «проклятьем размерности». Что это такое, можно представить, зафиксировав, например, $N=100$. Оценив величину $100!\approx 10^{158}$ и возможности развития вычислительной техники, приходим к выводу о том, что в обозримом будущем системы сотого порядка в принципе не могут быть решены по формулам Крамера. Заметим при этом, что, во-первых, метод Крамера будет неустойчив, т.е. погрешности округлений будут катастрофически нарастать, во-вторых, размерность $N = 100$ для современных задач не велика.

2.9. Матричный метод

Умножив слева обе части уравнения $\mathbf{Ax} = \mathbf{b}$ на $\mathbf{A}^{-1}$ ($\mathbf{A}^{-1}$ существует, если $\det\mathbf{A} \neq 0$), получим $\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$; $\mathbf{Ex} = \mathbf{A}^{-1}\mathbf{b}$, здесь $\mathbf{E}$ – единичная матрица. Следовательно, $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, т.е. чтобы найти решение системы N линейных уравнений с N неизвестными матричным методом, нужно матрицу, обратную матрице из коэффициентов системы, умножить на матрицу-столбец свободных членов. В результате получаем матрицу-столбец, которая и будет решением данной системы. Отыскание решения системы $\mathbf{Ax} = \mathbf{b}$ по формуле $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ называют матричным способом решения системы или решением по методу обратной матрицы.

Для решения СЛАУ матричным методом необходимо выполнить следующие действия:

1. Вычислить определитель $\det\mathbf{A}$. Если $\det\mathbf{A} \neq 0$, то матрица $\mathbf{A}^{-1}$ существует.

2. Составить союзную матрицу $\mathbf{A}^*$, элементами которой являются алгебраические дополнения элементов матрицы A:

$$\mathbf{A}^* = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \ldots \mathbf{A}_{1N} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \ldots \mathbf{A}_{2N} \\ . \; . \; . \; . \; . \; . \\ \mathbf{A}_{N1} & \mathbf{A}_{N2} \ldots \mathbf{A}_{NN} \end{pmatrix}$$

3. Транспонируя матрицу $\mathbf{A}^*$, получить матрицу $\tilde{\mathbf{A}}$, которую называют присоединенной матрицей:

$$\tilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{21} \ldots \mathbf{A}_{N1} \\ \mathbf{A}_{12} & \mathbf{A}_{22} \ldots \mathbf{A}_{N2} \\ . \; . \; . \; . \; . \; . \\ \mathbf{A}_{1N} & \mathbf{A}_{2N} \ldots \mathbf{A}_{NN} \end{pmatrix}.$$

4. Найти обратную матрицу $\mathbf{A}^{-1} = \tilde{\mathbf{A}} / \det \mathbf{A}$.

5. Найти решение системы $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

Пример 2.7. Решить СЛАУ с помощью матричного метода

$$\begin{pmatrix} 2 & 1 & 3 \\ 4 & -1 & 8 \\ -6 & 3 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 13 \\ 26 \\ 6 \end{pmatrix}$$

$$\det \mathbf{A} = \begin{vmatrix} 2 & 1 & 3 \\ 4 & -1 & 8 \\ -6 & 3 & 2 \end{vmatrix} = 2 \cdot (-1)^{(1+1)}[(-1 \cdot 2) - (3 \cdot 8)] +$$

$$+1 \cdot (-1)^{(1+2)}[(4 \cdot 2) - (-6 \cdot 8)] + 3 \cdot (-1)^{(1+3)}[(4 \cdot 3) - (-1 \cdot -6)] = -90;$$

Вычислим алгебраические дополнения

$\mathbf{A}_{11} = (-1)\cdot 2 - 3\cdot 8 = -26;$ $\mathbf{A}_{12} = -[4\cdot 2 - 8\cdot(-6)] = -56;$ $\mathbf{A}_{13} = 4\cdot 3 - (-1)\cdot(-6) = 6;$

$\mathbf{A}_{21} = -[1\cdot 2 - 3\cdot 3] = 7;$ $\mathbf{A}_{22} = 2\cdot 2 - 3\cdot(-6) = 22;$ $\mathbf{A}_{23} = -[2\cdot 3 - 1\cdot(-6)] = -12;$

$\mathbf{A}_{31} = 1\cdot 8 - 3\cdot(-1) = 11;$ $\mathbf{A}_{32} = -[2\cdot 8 - 3\cdot 4] = -4;$ $\mathbf{A}_{33} = 2\cdot(-1) - 4\cdot 1 = -6.$

Таким образом, союзная и присоединенная матрицы примут вид:

$$\mathbf{A}^* = \begin{pmatrix} -26 & -56 & 6 \\ 7 & 22 & -12 \\ 11 & -4 & -6 \end{pmatrix} \qquad \tilde{\mathbf{A}} = \begin{pmatrix} -26 & 7 & 11 \\ -56 & 22 & -4 \\ 6 & -12 & -6 \end{pmatrix}.$$

Далее вычислим обратную матрицу

$$\mathbf{A}^{-1} = \frac{1}{-90}\begin{pmatrix} -26 & 7 & 11 \\ -56 & 22 & -4 \\ 6 & -12 & -6 \end{pmatrix}$$

и найдем требуемое решение

$$\mathbf{x} = \frac{1}{-90}\begin{pmatrix} -26 & 7 & 11 \\ -56 & 22 & -4 \\ 6 & -12 & -6 \end{pmatrix}\begin{pmatrix} 13 \\ 26 \\ 6 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}.$$

2.10. Метод Гаусса

Наиболее известным и популярным способом решения линейных систем вида

$$\begin{array}{lllllll} a_{11}x_1 + & a_{12}x_2 + & a_{13}x_3 + & \ldots + & a_{1N}x_N & = & b_1, \\ a_{21}x_1 + & a_{22}x_2 + & a_{23}x_3 + & \ldots + & a_{2N}x_N & = & b_2, \\ a_{31}x_1 + & a_{32}x_2 + & a_{33}x_3 + & \ldots + & a_{3N}x_N & = & b_3, \\ \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots & \ldots\ldots & \ldots\ldots\ldots\ldots & \ldots & \ldots \\ a_{N1}x_1 + & a_{N2}x_2 + & a_{N3}x_3 + & \ldots + & a_{NN}x_N & = & b_N. \end{array} \qquad (2.19)$$

является метод последовательного исключения неизвестных – метод Гаусса[1] (GE[2]). Этот метод известен в различных вариантах, которые алгебраически тождественны, уже более 2000 лет. Методы отличаются характером хранения матриц, порядком исключения, способами предупреждения больших погрешностей округления и тем, как уточняются вычисленные решения. Имеются также варианты, специально приспособленные для систем с симметричными положительно определенными матрицами, которые хранятся в примерно вдвое меньшем объеме.

Вычисления с помощью метода Гаусса состоят из двух основных этапов, называемых прямым и обратным ходом. Прямой ход метода Гаусса заключается в последовательном исключении неизвестных из системы $\mathbf{Ax} = \mathbf{b}$ для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисление значений неизвестных производится на этапе обратного хода.

---

[1] Карл Фридрих Гаусс (1777–1855) – немецкий математик и физик, работы которого оказали большое влияние на дальнейшее развитие высшей алгебры, геометрии, теории чисел, теории электричества и магнетизма.

[2] Gaussian Elimination

2.10.1. *Схема единственного деления*

Рассмотрим сначала простейший вариант метода Гаусса, называемый схемой единственного деления.

Прямой ход состоит из $N-1$ шагов исключения.

1-й шаг. Целью этого шага является исключение неизвестного x_1 из уравнений с номерами $i = 2, 3, \ldots, N$. Предположим, что коэффициент $a_{11} \neq 0$. Будем называть его главным (или ведущим) элементом 1-го шага.

Найдем величины

$$t_{i1} = a_{i1} / a_{11} \ (i = 2, 3, \ldots, N), \tag{2.20}$$

называемые множителями 1-го шага. Вычтем последовательно из второго, третьего, …, N-го уравнений системы (2.19) первое уравнение, умноженное на $t_{21}, t_{31}, \ldots, t_{N1}$. Это позволит обратить в нуль коэффициенты при x_1 во всех уравнениях, кроме первого. В результате получим систему

$$\begin{aligned}
a_{11}x_1 &+ a_{12}x_2 &+ a_{13}x_3 &+ \ldots + a_{1N}x_N &= b_1,\\
&+ a^{(1)}_{22}x_2 &+ a^{(1)}_{23}x_3 &+ \ldots + a^{(1)}_{2N}x_N &= b^{(1)}_2,\\
&+ a^{(1)}_{32}x_2 &+ a^{(1)}_{33}x_3 &+ \ldots + a^{(1)}_{3N}x_N &= b^{(1)}_3,\\
&\ldots\ldots\ldots\ldots &\ldots\ldots\ldots\ldots &\ldots\ldots\ldots\ldots &\ldots\ldots\\
&+ a^{(1)}_{N2}x_2 &+ a^{(1)}_{N3}x_3 &+ \ldots + a^{(1)}_{NN}x_N &= b^{(1)}_N,
\end{aligned} \tag{2.21}$$

в которой $a^{(1)}_{ij}$ и $b^{(1)}_i$ $(i, j = 2, 3, \ldots, N)$ вычисляются по формулам

$$a^{(1)}_{ij} = a_{ij} - t_{i1}a_{1j}, \quad b^{(1)}_i = b_i - t_{i1}b_1. \tag{2.22}$$

2-й шаг. Целью этого шага является исключение неизвестного x_2 из уравнений с номерами $i = 2, 3, \ldots, N$. Пусть $a^{(1)}_{22} \neq 0$, где $a^{(1)}_{22}$ – коэффициент, называемый главным элементом 2-го шага. Вычислим множители

$$t_{i2} = a^{(1)}_{i2} / a^{(1)}_{22} \ (i = 3, 4, \ldots, N)$$

и вычтем последовательно из третьего, четвертого, …, N-го уравнений системы (2.21) второе уравнение, умноженное соответственно на $t_{32}, t_{42}, \ldots, t_{N2}$. В результате получим систему

$$\begin{aligned}
a_{11}x_1 &+ a_{12}x_2 &+ a_{13}x_3 &+ \ldots + a_{1N}x_N &= b_1,\\
&+ a^{(1)}_{22}x_2 &+ a^{(1)}_{23}x_3 &+ \ldots + a^{(1)}_{2N}x_N &= b^{(1)}_2,\\
& &+ a^{(2)}_{33}x_3 &+ \ldots + a^{(2)}_{3N}x_N &= b^{(2)}_3,\\
& &\ldots\ldots\ldots\ldots &\ldots\ldots\ldots\ldots &\ldots\ldots\\
& &+ a^{(2)}_{N3}x_3 &+ \ldots + a^{(2)}_{NN}x_N &= b^{(2)}_N,
\end{aligned} \tag{2.23}$$

где $a^{(2)}_{ij}$ и $b^{(2)}_i$ $(i, j = 3, 4, \ldots, N)$ вычисляются по формулам

$$a^{(2)}_{ij} = a^{(1)}_{ij} - t_{i2}a^{(1)}_{2j}, \quad b^{(2)}_i = b^{(1)}_i - t_{i2}b^{(1)}_2.$$

Аналогично проводятся остальные шаги. Опишем очередной k-й шаг.

k-й шаг. В предположении, что главный (ведущий) элемент k-го шага $a_{kk}^{(k-1)}$ отличен от нуля, вычислим множители k-го шага

$$t_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)} \quad (i = k+1, \ldots, N)$$

и вычтем последовательно из $(k+1)$-го, …, N-го уравнений полученной на предыдущем шаге системы k-е уравнение, умноженное соответственно на $t_{k+1,k}$, $t_{k+1,k}$, …, t_{Nk}.

После $(N-1)$-го шага исключения получим систему уравнений

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \ldots & + & a_{1N}x_N & = & b_1, \\ & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & \ldots & + & a_{2N}^{(1)}x_N & = & b_2^{(1)}, \\ & & & & a_{33}^{(2)}x_3 & + & \ldots & + & a_{3N}^{(2)}x_N & = & b_3^{(2)}, \\ & & & & & & \ldots\ldots & & \ldots\ldots\ldots & \ldots & \ldots \\ & & & & & & & & a_{NN}^{(N-1)}x_N & = & b_N^{(N-1)}, \end{array} \qquad (2.24)$$

матрица $\mathbf{A}^{(N-1)}$ которой является верхней треугольной.

Обратный ход. Из последнего уравнения системы (2.24) находим x_N. Подставляя найденное значение x_N в последнее уравнение, получаем x_{N-1}. Осуществляя обратную подстановку, далее последовательно находим x_{N-2}, x_{N-3}, …, x_1. Вычисления неизвестных здесь проводятся по формулам

$$x_N = b_N^{(N-1)} / a_{NN}^{(N-1)},$$

$$x_k = \frac{b_k^{(k-1)} - a_{k,k+1}^{(k-1)}x_{k+1} - \ldots - a_{kN}^{(k-1)}x_N}{a_{kk}^{(k-1)}} \quad (k = N-1, \ldots, 1). \qquad (2.25)$$

Трудоемкость метода.

Вычисления 1-го шага исключения по формулам (2.20), (2.22) требуют выполнения $N-1$ деления, $(N-1)N$ умножений и $(N-1)N$ вычитаний, т.е. общее число арифметических операций составляет $Q_1 = 2(N-1)^2 + 3(N-1)$. Аналогично, на 2-м шаге требуется $Q_2 = (N-2)^2 + 3(N-2)$ операций, а на k-м шаге – $Q_k = 2(N-k)^2 + 3(N-k)$ операций.

Подсчитаем теперь приближенно общее число Q арифметических операций прямого хода, считая N достаточно большой:

$$Q = \sum_{k=1}^{N-1} Q_k = 2\sum_{k=1}^{N-1}(N-k)^2 + 3\sum_{k=1}^{N-1}(N-k) = 2\sum_{k=1}^{N-1}k^2 + 3\sum_{k=1}^{N-1}k =$$

$$= \frac{2(N-1)N(2N-1)}{6} + \frac{3(N-1)N}{2} \approx \frac{2}{3}N^3.$$

Как нетрудно видеть, для реализации обратного хода по формулам (2.25) нужно всего N^2 операций, что при больших N пренебрежимо мало

по сравнению с числом операций прямого хода. Таким образом, для реализации метода Гаусса требуется примерно $(2/3)N^3$ арифметических операций, причем подавляющее число этих действий совершается на этапе прямого хода.

Далее приведен простой алгоритм решения СЛАУ методом Гаусса.

Алгоритм метода Гаусса

Для $k = 1, 2, \ldots, N-1$

  Для $i = k+1, \ldots, N$

    $t_{ik} = a_{ik} / a_{kk}$

    $b_i = b_i - t_{ik}\, b_k$

  Увеличить i

  Для $j = k+1, \ldots, N$

    $a_{ij} = a_{ij} - t_{ik}\, a_{kj}$

  Увеличить j

Увеличить k

$x_N = b_N / a_{NN}$

Для $k = N-1, \ldots, 2, 1$

$$x_k = \left(b_k - \sum_{j=k+1}^{N} a_{kj} x_j \right) / a_{kk}$$

Увеличить k

Подав на вход алгоритма квадратную матрицу коэффициентов и вектор свободных членов и выполнив три вложенных цикла вычислений прямого хода (строки 1 – 6), приведем исходную систему к треугольному виду. Выполнив цикл вычислений обратного хода (строки 7 – 9) на выходе алгоритма получим точное решение системы (в обратном порядке), если, разумеется, ни один из знаменателей не обращается в нуль и все вычисления проводятся точно.

Пример 2.8. Методом Гаусса решить систему

$$\begin{aligned} 10x_1 &+ 6x_2 &+ 2x_3 &+ &&= 25, \\ 5x_1 &+ x_2 &- 2x_3 &+ 4x_4 &&= 14, \\ 3x_1 &+ 5x_2 &- x_3 &- x_4 &&= 10, \\ &6x_2 &- 2x_3 &+ 2x_4 &&= 8. \end{aligned} \qquad (2.26)$$

Прямой ход.

1-й шаг. Вычислим множители $t_{21} = a_{21} / a_{11} = 5 / 10 = 0.5$, $t_{31} = a_{31} / a_{11} = 3 / 10 = 0.3$, $t_{41} = a_{41} / a_{11} = 0 / 10 = 0$. Вычитая из второго, третьего и четвертого уравнений системы (2.26) первое уравнение, умноженное на t_{21}, t_{31} и t_{41} соответственно, получаем:

$$\begin{aligned} 10x_1 &+ 6x_2 &+ 2x_3 & & &= 25,\\ &- 2x_2 &- 3x_3 &+ 4x_4 &= 1.5,\\ & 3.2x_2 &+ 0.4x_3 &- x_4 &= 2.5,\\ & 6x_2 &- 2x_3 &+ 2x_4 &= 8. \end{aligned} \qquad (2.27)$$

2-й шаг. Вычислим множители $t_{32} = a_{32}^{(1)} / a_{22}^{(1)} = 3.2/(-2) = -1.6$, $t_{42} = 6/(-2) = -3$. Вычитая из третьего и четвертого уравнений системы (2.27) второе уравнение, умноженное на t_{32}, t_{42} соответственно, приходим к системе

$$\begin{aligned} 10x_1 &+ 6x_2 &+ 2x_3 & & &= 25,\\ &- 2x_2 &- 3x_3 &+ 4x_4 &= 1.5,\\ & &- 4.4x_3 &+ 5.4x_4 &= 4.9,\\ & &- 11x_3 &+ 14x_4 &= 12.5. \end{aligned} \qquad (2.28)$$

3-й шаг. Вычисляя множитель $t_{43} = (-11) / (-4.4) = 2.5$ и вычитая из четвертого уравнения системы (2.28) третье уравнение, умноженное на t_{43}, приводим систему к треугольному виду:

$$\begin{aligned} 10x_1 &+ 6x_2 &+ 2x_3 & & &= 25,\\ &- 2x_2 &- 3x_3 &+ 4x_4 &= 1.5,\\ & &- 4.4x_3 &+ 5.4x_4 &= 4.9,\\ & & & 0.5x_4 &= 0.25. \end{aligned} \qquad (2.29)$$

Обратный ход. Из последнего уравнения системы находим $x_4 = 0.5$. Подставляя значение x_4 в третье уравнение, находим $x_3 = (4.9 - 5.4x_4) / (-4.4) = -0.5$. Продолжая далее обратную подстановку, получаем $x_2 = (1.5 + 3x_3 - 4x_4) / (-2) = 1$, $x_1 = (25 - 6x_2 - 2x_3) / 10 = 2$. Итак, $x_1 = 2$, $x_2 = 1$, $x_3 = -0.5$, $x_4 = 0.5$.

Результаты вычислений приведены в табл. 2.3.

Необходимость выбора главных элементов. Заметим, что вычисление множителей, а также обратная подстановка предполагают деление на главные элементы $a_{kk}^{(k-1)}$. 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.

Таблица 2.3

Наименование	a_{i1}	a_{i2}	a_{i13}	a_{i4}	b_i	t_{ij}, x_i
Исходная система	10	6	2	0	25	–
	5	1	–2	4	14	–
	3	5	1	–1	10	–
	0	6	–2	2	8	–
1-й шаг прямого хода	10	6	2	0	25	–
	0	–2	–3	4	1.5	0.5
	0	3.2	0.4	–1	2.5	0.3
	0	6	–2	2	8	0
2-й шаг прямого хода	10	6	2	0	25	–
	0	–2	–3	4	1.5	–
	0	0	–4.4	5.4	4.9	1.6
	0	0	–11	14	7.5	–3
3-й шаг прямого хода и обратный ход	10	6	2	0	25	2
	0	–2	–3	4	1.5	1
	0	0	–4.4	5.4	4.9	–0.5
	0	0	0	0.5	0.25	0.5

Пример 2.9. Используя метод Гаусса, решим систему уравнений

$$\begin{aligned} 2x_1 &- 9x_2 + 5x_3 = -4, \\ 1.2x_1 &- 5.3999x_2 + 6x_3 = 0.6001, \\ x_1 &- x_2 - 7.5x_3 = -8.5 \end{aligned} \qquad (2.30)$$

на 6-разрядном десятичном компьютере.

Прямой ход.

1-й шаг. Вычисляем множители $t_{21} = 0.6$ и $t_{31} = 0.5$ и преобразуем систему к виду

$$\begin{aligned} 2x_1 &- 9x_2 + 5x_3 = -4, \\ &0.0001x_2 + 3x_3 = 3.0001, \\ &3.5x_2 - 10x_3 = -6.5. \end{aligned} \qquad (2.31)$$

Все вычисления на этом шаге выполняются без округлений.

2-й шаг. После вычисления множителя $t_{32} = 3.5 / 0.0001 = 35000$ последнее уравнение системы должно быть преобразовано к виду $a_{33}^{(2)} x_3 = b_3^{(2)}$, где $a_{33}^{(2)} = -10 - 3 \cdot t_{32} = -105010$, $b_3^{(2)} = -6.5 - 3.0001 = -105010$. Однако на используемом компьютере будет получено уравнение

$$-105010x_3 = -105011. \qquad (2.32)$$

Действительно, коэффициент $a_{33}^{(2)}$ определяется точно, так как при его вычислении не возникает чисел, мантиссы которых имеют более шести разрядов. В то же время при вычислении $b_3^{(2)}$ умножение коэффициента

3.0001 на t_{32} дает 7-разрядное число 105003.5, после округления последнего числа до шести разрядов получится 105004. Вычисление $b_3^{(2)}$ завершается операцией вычитания: $b_3^{(2)} \approx -6.5 - 105004 = -105010.5$. После округления последнего числа до шести разрядов мантиссы приходим к уравнению (2.32).

Обратный ход. Из уравнения (2.32) находим $x_3 \approx 1.00001$. Сравнение с истинным значением $x_3 = 1$ показывает, что эта величина получена с очень высокой для используемого компьютера точностью.

Дальнейшие вычисления дают

$$x_2 = (3.0001 - 3x_3) / 0.0001 = (3.0001 - 3.00003) / 0.0001 = 0.7;$$

$$x_1 = (-4 + 9x_2 - 5x_3) / 2 = (-4 + 6.3 - 5.00005) / 2 = -1.350025.$$

После округления имеем $x_1 = -1.35003$.

Как нетрудно видеть, найденные значения неизвестных имеют мало общего с истинными значениями решения $x_1 = 0$, $x_2 = 1$, $x_3 = 1$.

В чем же причина появления такой значительной погрешности? Говорить о накоплении ошибок округления не приходится, так как всего было выполнено 28 арифметических операций и лишь в четырех случаях требовалось округление. Предположение о плохой обусловленности не подтверждается; вычисление $\mathrm{cond}_\infty(\mathbf{A})$ дает значение, равное примерно 100.

В действительности причина состоит в использовании на 2-м шаге малого ведущего элемента $a_{22}^{(2)} = 0.0001$. Следствием этого стало появление большого множителя t_{32} и существенное возрастание коэффициентов в последнем уравнении системы.

Таким образом, изложенный вариант метода Гаусса оказался некорректным и, следовательно, непригодным для вычислений на компьютере. Этот метод может привести к аварийному останову (если $a_{kk}^{(k-1)} = 0$ при некотором k), и вычисления могут оказаться неустойчивыми.

2.10.2. *Метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора)*

Описание метода. На k-м шаге прямого хода коэффициенты уравнений системы с номерами $i = k + 1, \ldots, N$ преобразуются по формулам

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - t_{ik} a_{kj}^{(k-1)},\ b_i^{(k)} = b_i^{(k-1)} - t_{ik} a_k^{(k-1)},\ i = k+1,\ \ldots,\ N. \qquad (2.33)$$

Интуитивно ясно, что во избежание сильного роста коэффициентов системы и связанных с этим погрешностей нельзя допускать появления больших множителей t_{ik}.

В методе Гаусса с выбором главного элемента[1] по столбцу гарантируется, что $|t_{ik}| \le 1$ для всех $k = 1, 2, \ldots, N-1$ и $i = k+1, \ldots, N$. Отличие этого варианта метода Гаусса от схемы единственного деления состоит в том, что на k-м шаге исключения в качестве главного элемента выбирают максимальный по модулю коэффициент $a_{i_k k}^{(k-1)}$ при неизвестной x_k в уравнениях с номерами $i = k, k+1, \ldots, N$. Затем соответствующее выбранному коэффициенту уравнение с номером i_k меняют местами с k-м уравнением системы для того, чтобы главный элемент занял место коэффициента $a_{kk}^{(k-1)}$.

После этой перестановки исключение неизвестного x_k производят, как в схеме единственного деления.

Частичное упорядочивание по столбцам требует внесения в алгоритм следующих изменений: между строками 1 и 2 нужно сделать вставку:

«Найти $m \ge k$ такое, что $|a_{mk}| = \max\limits_{i \ge k} |a_{ik}|$; если $a_{mk} = 0$, остановить работу алгоритма («однозначного решения нет»), иначе поменять местами b_k и b_m, a_{kj} и a_{mj} при всех $j = k, \ldots, N$»

Пример 2.10. Решим СЛАУ (2.30) методом Гаусса с выбором главного элемента по столбцу на 6-разрядном десятичном компьютере.

Прямой ход.

1-й шаг. Максимальный в первом столбце элемент матрицы находится в первой строке, поэтому перестановка уравнений не нужна. Здесь 1-й шаг проводится точно так же, как и в примере 2.9.

2-й шаг. Среди элементов $a_{22}^{(1)} = 0.0001$ и $a_{32}^{(1)} = 3.5$ матрицы системы (2.31) максимальный по модулю принадлежит третьему уравнению. Меняя местами, второе и третье уравнения, получаем систему

$$\begin{aligned} 2x_1 - 9x_2 + 5x_3 &= -4, \\ 3.5x_2 - 10x_3 &= -6.5, \\ 0.0001x_2 + 3x_3 &= 3.0001. \end{aligned}$$

После вычисления $t_{32} = 0.0001 / 3.5 \approx 2.85714 \cdot 10^{-5}$ последнее уравнение системы преобразуется к виду $3.00029x_3 = 3.00029$.

Обратный ход. Из последнего уравнения находим $x_3 = 1$. Далее, имеем $x_2 = (-6.5 + 10x_3) / 3.5 = 1$, $x_1 = (-4 + 9x_2 - 5x_3) / 2 = (-4 + 9 - 5) / 2 = 0$. В данном случае ответ получается точным.

[1] Выбор главного элемента иногда называют пивотированием (от англ. pivoting)

Заметим, что дополнительная работа по выбору главных элементов в схеме частичного выбора требует порядка N^2 действий, что практически не влияет на общую трудоемкость метода.

Вычислительная устойчивость схемы частичного выбора. Детальное исследование метода Гаусса показывает, что действительной причиной неустойчивости схемы единственного деления является возможность неограниченного роста элементов промежуточных матриц $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$, …, $\mathbf{A}^{(N-1)}$ в процессе прямого хода. Так как на k-м шаге схемы частичного выбора $|t_{ik}| \le 1$, то для вычисленных по формулам (2.33) элементов $a_{ij}^{(k)}$ справедлива оценка $|a_{ij}^{(k)}| \le |a_{ij}^{(k-1)}| + |a_{kj}^{(k-1)}|$. Следовательно, максимальное по модулю значение элементов матрицы возрастает на одном шаге не более чем в 2 раза и в самом неблагоприятном случае $N - 1$ шаг прямого хода даст коэффициент роста $\varphi(N) = 2^{N-1}$.

Гарантия ограниченности роста элементов матрицы делает схему частичного выбора вычислительно устойчивой. Более того, для нее оказывается справедливой следующая оценка погрешности:

$$\delta(\mathbf{x}^*) \le f(N)\cdot \mathrm{cond}_E(\mathbf{A})\cdot \varepsilon_M. \qquad (2.34)$$

Здесь $\mathbf{x}^*$ – вычисленное на компьютере решение системы; $\delta(\mathbf{x}^*) = \|\mathbf{x} - \mathbf{x}^*\|_2 / \|x\|_2$ – его относительная погрешность; $\mathrm{cond}_E(\mathbf{A}) = \|\mathbf{A}\|_E\|\mathbf{A}^{-1}\|_E$ – число обусловленности матрицы $\mathbf{A}$; ε_M – машинное эпсилон; наконец, $f(N) = C(N)\,\varphi(N)$, где $C(N)$ – некоторая медленно растущая функция, зависящая от порядка N системы (типа степенной функции с небольшим показателем).

Наличие в оценке (2.34) множителя $\varphi(N) = 2^{N-1}$ указывает на то, что при большом N схема частичного выбора может оказаться плохо обусловленной и возможна существенная потеря точности. Однако практика матричных вычислений показывает, что существенный рост элементов матрицы происходит крайне редко. В подавляющем большинстве случаев действительное значение коэффициента роста не превышает 8–10. Если система хорошо обусловлена, то погрешность вычисленного решения оказывается, как правило, малой.

Иногда для проверки качества приближенного решения $\mathbf{x}^*$ вычисляют невязку $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}^*$ и о степени близости приближенного решения к точному пытаются судить по тому, насколько мала невязка. Этот метод ненадежен по отношению к схеме частичного выбора, так как известно, что она гарантированно дает малые невязки. Более точно это утверждение можно сформировать так: справедлива оценка

$$\|\mathbf{r}\|_2 \le f(N)\|\mathbf{A}\|_E\cdot\|\mathbf{x}\|_2\cdot\varepsilon_M, \qquad (2.35)$$

где $f(N)$ то же, что и в оценке (2.34). Заметим, что в неравенство (2.35) не входит число обусловленности.

Аналогично методу Гаусса с постолбцовым выбором главного элемента можно реализовать алгоритм с построчным выбором ведущего элемента.

2.10.3. *Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора)*

В этой схеме допускается нарушение естественного порядка исключения неизвестных. На 1-м шаге метода среди элементов a_{ij} определяют максимальный по модулю элемент $a_{i_1 j_1}$. Первое уравнение системы и уравнение с номером i_1 меняют местами. Далее стандартным образом производят исключение неизвестного x_{j_1} из всех уравнений, кроме первого.

На k-м шаге метода среди коэффициентов $a_{ij}^{(k-1)}$ при неизвестных в уравнениях системы с номерами $i = k, \ldots, N$ выбирают максимальный по модулю коэффициент $a_{i_k j_k}^{(k-1)}$. Затем k-е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное x_{j_k} из уравнений с номерами $i = k + 1, \ldots, N$. На этапе обратного хода неизвестные вычисляют в следующем порядке: $x_{j_N}, x_{j_{N-1}}, \ldots, x_{j_1}$.

Пример 2.11. Решим систему (2.30), используя схему полного выбора на 6-разрядном десятичном компьютере.

Прямой ход.

1-й шаг. Максимальный по модулю элемент a_{12} содержится в первом уравнении, поэтому перестановка уравнений не нужна. Исключаем x_2 из второго и третьего уравнений, используя $t_{21} = -5.3999 / (-9) \approx 0.5999989$ и $t_{31} = -1 / (-9) \approx 0.111111$. В результате получаем систему

$$\begin{aligned} 2x_1 &- 9x_2 + 5x_3 &&= -4, \\ 0.000022x_1 &+ 3.00006x_3 &&= 3.00006, \\ 0.777778x_1 &- 8.05556x_3 &&= -8.05556. \end{aligned}$$

2-й шаг. Среди коэффициентов при неизвестных во втором и третьем уравнениях максимальным по модулю является коэффициент $a_{33}^{(1)} = -8.05556$. Переставляя местами второе и третье уравнения и исключая неизвестное x_3 (соответствующий множитель $t_{32} = 3.00006 / (-8.05556) \approx -0.372421$), приходим к системе

$$
\begin{aligned}
2x_1 \quad &- \quad 9x_2 \quad + \quad 5x_3 \quad = \quad -4, \\
0.777778x_1 \quad &- \quad 8.05556x_3 \quad = \quad 3.00006, \\
0.289683x_1 \quad &= \quad 2.89240\cdot 10^{-7}.
\end{aligned}
$$

Обратный ход. Из последнего уравнения находим $x_1 = 2.89240\cdot 10^{-7} / 0.289683 \approx 9.98470\cdot 10^{-7}$. Далее, имеем $x_3 = (-8.05556 - 0.777778x_1) / (-8.05556) \approx 1.00000$, $x_2 = (-4 - 2x_1 - 5x_3) / (-9) \approx 1.00000$.

Округляя найденные значения до пяти цифр после десятичной точки, получим ответ: $x_1 = 0.00000$, $x_2 = 1.00000$, $x_3 = 1.00000$. Заметим, что в данном случае получено решение, совпадающее с точным.

Схема полного выбора по сравнению со схемой частичного выбора дает существенное замедление роста элементов матрицы. Доказано, что для нее коэффициент роста $\varphi(N)$, входящий в оценку (2.34), не превышает значения $N^{1/2}\,(2^{1}\cdot 3^{1/2}\cdot 4^{1/3}\ldots N^{1/(N-1)})^{1/2} \le 1.8N^{0.25\ln N}$ (что меньше значения $\varphi(N) = 2^{N-1}$ для схемы частичного выбора). Более того, длительное время существовала гипотеза Уилкинсона, согласно которой для схемы полного выбора $\varphi(N) \le N$. Только в 1991 г. Была найдена матрица 13-го порядка, для которой $\varphi(N) > 13$. Таким образом, для хорошо обусловленных систем этот вариант метода Гаусса является хорошо обусловленным.

Однако гарантия хорошей обусловленности достигается здесь ценой значительных затрат на выбор главных элементов. Для этого дополнительно к $(2/3)N^3$ арифметическим действиям требуется произвести примерно $N^3/3$ операций сравнения, что может ощутимо замедлить процесс решения задачи на компьютере. Поэтому в большинстве случаев на практике предпочтение отдается все же схеме частичного выбора. Как уже отмечено выше, ситуации, когда при использовании этого варианта метода Гаусса происходит существенный рост элементов, встречаются чрезвычайно редко. Более того, эти ситуации могут быть легко выявлены с помощью заложенных в современных программах эффективных методов слежения за ростом элементов матриц.

2.10.4. *Случай, когда выбор главных элементов не нужен*

Известно что, для некоторых классов матриц при использовании схемы единственного деления главные элементы гарантированно располагаются на главной диагонали и потому применять частичный выбор нет необходимости. Так, например, обстоит дело для систем с симметричными положительно определенными матрицами, а также с матрицами, обладающими свойством строчного диагонального преобладания

$$\sum_{\substack{j=1 \\ j\neq i}}^{N} |a_{ij}| < |a_{ii}|, \quad i = 1, 2, ..., N. \tag{2.36}$$

Матрицы, удовлетворяющие условию (2.36), таковы, что в каждой из строк модуль элемента a_{ii}, расположенного на главной диагонали, больше суммы модулей остальных элементов строки.

2.10.5. *Масштабирование*

Перед началом решения целесообразно масштабировать систему так, чтобы ее коэффициенты были величинами порядка единицы.

Существуют два естественных способа масштабирования системы **Ax** = **b**. Первый заключается в умножении каждого из уравнений на некоторый масштабирующий множитель t_i. Второй состоит в умножении на масштабирующий множитель α_j каждого j-го столбца матрицы, что соответствует замене переменных $x'_j = \alpha_j^{-1} x_j$ (фактически – это замена единиц измерения). В реальных ситуациях чаще всего масштабирование может быть выполнено без существенных трудностей. Однако подчеркнем, что в общем случае удовлетворительного способа масштабирования пока не найдено.

На практике масштабирование обычно производят с помощью деления каждого уравнения на его наибольший по модулю коэффициент. Это вполне удовлетворительный способ для большинства реально встречающихся задач.

2.11. LU-разложение матриц

Пусть $\mathbf{A} = \left(a_{ij}\right)_{i,j=1}^{N}$ – данная $N \times N$-матрица, а $\mathbf{L} = \left(l_{ij}\right)_{i,j=1}^{N}$ и $\mathbf{U} = \left(u_{ij}\right)_{i,j=1}^{N}$ – соответственно нижняя (левая) и верхняя (правая) треугольные матрицы. Справедливо следующее утверждение. Если все главные миноры квадратной матрицы **A** отличны от нуля, то существуют такие нижняя **L** и верхняя **U** треугольные матрицы, что **A** = **LU** (главными минорами матрицы $\mathbf{A} = \left(a_{ij}\right)_{i,j=1}^{N}$ называются определители подматриц $\mathbf{A}_k = \left(a_{ij}\right)_{i,j=1}^{k}$, где k = 1, 2,…, N – 1). Если элементы диагонали одной из матриц, **L** или **U**, фиксированы (ненулевые), то такое разложение единственно.

Реализация LU-разложения с фиксированием диагонали верхней треугольной матрицы (u_{ij} = 1 при $i = j$) называется методом Краута. Рассмот-

рим, более часто используемое на практике, разложение матриц при фиксировании диагонали нижней треугольной матрицы ($l_{ij} = 1$ при $i = j$) – метод Дулитла. Находят l_{ij} при $i > j$ ($l_{ij} = 0$ при $i < j$) и u_{ij} при $i \le j$ ($u_{ij} = 0$ при $i > j$) такие, чтобы

$$\begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{N1} & l_{N2} & \dots & 1 \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1N} \\ 0 & u_{22} & \dots & u_{1N} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{NN} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{1N} & a_{2N} & \dots & a_{NN} \end{pmatrix}.$$

Выполнив перемножение матриц, на основе поэлементного приравнивания левых и правых частей приходим к $N \times N$ матрице уравнений

$$\begin{array}{llll} u_{11} = a_{11}, & u_{12} = a_{12}, & \dots, & u_{1N} = a_{1N}, \\ l_{21}u_{11} = a_{21}, & l_{21}u_{12} + u_{22} = a_{22}, & \dots, & l_{21}u_{1N} + u_{2N} = a_{2N}, \\ \dots. & \dots. & \dots. & \dots. \\ l_{N1}u_{11} = a_{N1}, & l_{N1}u_{12} + l_{N2}u_{22} = a_{N2}, & \dots, & l_{N1}u_{1N} + \dots + u_{NN} = a_{NN}, \end{array}$$

относительно $N \times N$-матрицы неизвестных

$$\begin{array}{l} u_{11}, u_{12}, \dots, u_{1N} \\ l_{21}, u_{22}, \dots, u_{2N} \\ \dots\dots\dots\dots\dots \\ l_{N1}, l_{N2}, \dots, u_{NN} \end{array} \qquad (2.37)$$

Легко видеть, что все отличные от 0 и 1 элементы матриц **L** и **U** могут быть однозначно вычислены с помощью всего двух формул

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \quad (\text{где } i \le j), \qquad (2.38)$$

$$l_{ij} = \frac{1}{u_{jj}}\left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \right) \quad (\text{где } i > j). \qquad (2.39)$$

При внимательном рассмотрении приведенных преобразований можно заметить, что реализовать LU-разложение по данным формулам можно различными методами, например, построчным вычислением, т.е. пока не вычислена i-я строка матриц **L** и **U**, алгоритм не модернизирует $i + 1$ строку.

При практическом выполнении разложения (факторизации) матрицы **A** нужно иметь в виду следующие два обстоятельства. Во-первых, организация вычислений по формулам (2.38)–(2.39) должна предусматривать переключение счета с одной формулы на другую. Это удобно делать, ориентируясь на матрицу неизвестных (2.37) (ее, кстати, можно интер-

претировать как N^2-мерный массив для компактного хранения LU-разложения в памяти компьютера), а именно, первая строка (2.37) вычисляется по формуле (2.38) при $i = 1$, $j = 1, 2,\ldots, N$; первый столбец (2.37) (без первого элемента) – по формуле (2.39) при $j = 1$, $i = 2,\ldots, N$, и т.д. Во-вторых, препятствием для осуществимости описанного процесса LU-разложения матрицы **A** может оказаться равенство нулю диагональных элементов матрицы **U**, поскольку на них выполняется деление в формуле (2.39). Отсюда следует требование, накладываемое на главные миноры. Для определенных классов матриц требования о разложении заведомо выполняются. Это относится, например, к матрицам с преобладанием диагональных элементов.

Далее приведена построчная схема LU-разложения (так называемая *ikj*-версия), которая является, пожалуй, самой предпочтительной для программной реализации [8].

Алгоритм LU-разложения (*ikj*-версия)

Для $i = 2, \ldots, N$
  Для $k = 1, \ldots, i - 1$
    $a_{ik} = a_{ik} / a_{kk}$
    Для $j = k + 1, \ldots, N$
      $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
    Увеличить j
  Увеличить k
Увеличить i

Данный алгоритм позволяет пересчитать i-ю строку матрицы **A** в i-ю строку матриц **L** и **U**. Каждые $1, 2,\ldots, j - 1$ строки участвуют в определении j-й строки матриц **L** и **U**, но сами больше не модернизируются.

Существует порядка десяти вариантов осуществления LU-разложения [8]. Также иногда данное представление матрицы в виде произведения матриц называют LR-разложением.

Пример 2.12. Проиллюстрируем LU-разложение на примере решения системы (2.26). На основании данных табл. 2.3 можно записать

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.3 & -1.6 & 1 & 0 \\ 0 & -3 & 2.5 & 1 \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 10 & 6 & 2 & 0 \\ 0 & -2 & -3 & 4 \\ 0 & 0 & -4.4 & 5.4 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}.$$

Следовательно, LU-разложение матрицы системы имеет вид

$$\begin{pmatrix} 10 & 6 & 2 & 0 \\ 5 & 1 & -2 & 4 \\ 3 & 5 & 1 & -1 \\ 0 & 6 & -2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0.3 & -1.6 & 1 & 0 \\ 0 & -3 & 2.5 & 1 \end{pmatrix} \cdot \begin{pmatrix} 10 & 6 & 2 & 0 \\ 0 & -2 & -3 & 4 \\ 0 & 0 & -4.4 & 5.4 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}.$$

2.12. Решение СЛАУ с помощью LU-разложения

Если матрица **A** исходной системы **Ax** = **b** разложена в произведение треугольных **L** и **U**, то вместо **Ax** = **b** можно записать

$$\mathbf{LUx} = \mathbf{b}.$$

Введя вектор вспомогательных переменных **y**, последнее выражение можно переписать в виде системы

$$\begin{cases} \mathbf{Ly} = \mathbf{b}, \\ \mathbf{Ux} = \mathbf{y}. \end{cases}$$

Таким образом, решение данной системы с квадратной матрицей коэффициентов свелось к последовательному решению двух систем с треугольными матрицами коэффициентов.

Очевидно, все y_i могут быть найдены из системы **Ly** = **b** при $i = 1, 2, \ldots, N$ по формуле (прямой ход)

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k. \tag{2.40}$$

Значения неизвестных x_i находятся из системы **Ux** = **y** в обратном порядке, т.е. при $i = N, N-1, \ldots, 1$, по формуле (обратный ход)

$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^{N} u_{ik} x_k \right). \tag{2.41}$$

Итак, решение СЛАУ посредством LU-факторизации сводится к организации вычислений по четырем формулам: совокупности формул (2.38) – (2.39) для получения матрицы **L** + **U** – **I** (2.37) ненулевых и неединичных элементов матриц для **L** и **U**, формулы (2.40) для получения вектора свободных членов треугольной системы **Ux** = **y** и формулы (2.41), генерирующей решение исходной системы **Ax** = **b**.

Пример 2.13. Решим систему

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 + &= 8, \\ 5x_1 + x_2 - 2x_3 + 4x_4 &= 7, \\ 3x_1 + 5x_2 - x_3 - x_4 &= 2, \\ 6x_2 - 2x_3 + 2x_4 &= 2. \end{aligned}$$

Воспользуемся LU-разложением матрицы системы, указанным в примере 2.12. Сначала преобразуем вектор правой части $\mathbf{b} = (8, 7, 2, 2)^T$ по формулам прямого хода.

1-й шаг. $b_2^{(1)} = b_2 - t_{21}b_1 = 7 - 0.5 \cdot 8 = 3$, $b_3^{(1)} = b_3 - t_{31}b_1 = 2 - 0.3 \cdot 8 = -0.4$, $b_4^{(1)} = b_4 - t_{41}b_1 = 2 - 0 \cdot 8 = 2$. После 1-го шага получим $\mathbf{b}^{(1)} = (8, 3, -0.4, 2)^T$.

2-й шаг. $b_3^{(2)} = b_3^{(1)} - t_{32}b_2^{(1)} = -0.4 - (-1.6) \cdot 3 = 4.4$, $b_4^{(2)} = b_4^{(1)} - t_{42}b_2^{(1)} = 2 - (-3) \cdot 3 = 11$. После 2-го шага найдем $\mathbf{b}^{(2)} = (8, 3, 4.4, 11)^T$.

3-й шаг. $b_4^{(3)} = b_4^{(2)} - t_{43}b_3^{(2)} = 11 - 2.5 \cdot 4.4 = 0$. В результате прямого хода получен вектор $\mathbf{b}^{(3)} = (8, 3, 4.4, 0)^T$ и система приведена к виду

$$\begin{aligned} 10x_1 + 6x_2 + 2x_3 &= 8, \\ -2x_2 - 3x_3 + 4x_4 &= 3, \\ -4.4x_3 + 5.4x_4 &= 4.4, \\ 0.5x_4 &= 0. \end{aligned}$$

Обратный ход дает значения $x_4 = 0$, $x_3 = -1$, $x_2 = 0$, $x_1 = 1$.

Обратим внимание на тот факт, что выполнение расчетов по формулам (2.38) – (2.40) можно интерпретировать как преобразование системы $\mathbf{Ax} = \mathbf{b}$ к системе $\mathbf{Ux} = \mathbf{y}$. С системой, являющейся результатом прямого хода метода Гаусса, последняя имеет не только структурное сходство, но полностью совпадает с ней. Совпадение первых уравнений очевидно, коэффициенты при неизвестных и свободные члены вторых уравнений легко выражаются одинаково через исходные данные. Обычно идентичность этих систем показывают на основе представления прямого хода метода Гаусса как последовательности умножений матрицы $\mathbf{A}$ слева на матрицы очень простой структуры, такой, что в итоге получается матрица $\mathbf{U}$, а последовательность обратных преобразований дает $\mathbf{L}$. Рассмотрим это представление подробнее.

Вернемся еще раз к методу Гаусса, чтобы рассмотреть его с более общих позиций. Излагаемый ниже подход оказался чрезвычайно плодотворным и привел не только к более глубокому пониманию метода, но и позволил создать высокоэффективные машинные алгоритмы его реализации, а также рассмотреть другие точные методы с единой точки зрения.

Рассмотрим сначала простейший вариант метода Гаусса.

При выполнении вычислений 1-го шага исключения по схеме единственного деления система уравнений приводится к виду

$$\mathbf{A}^{(1)}\mathbf{x} = \mathbf{b}^{(1)}, \qquad (2.42)$$

$$\mathbf{A}^{(1)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2N}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3N}^{(1)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & a_{N2}^{(1)} & a_{N3}^{(1)} & \dots & a_{NN}^{(1)} \end{pmatrix}, \quad \mathbf{b}^{(1)} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(1)} \\ \dots \\ b_N^{(1)} \end{pmatrix},$$

а коэффициенты $a_{ij}^{(1)}$, $b_i^{(1)}$ $(i, j = 2, 3, ..., N)$ вычисляются по формулам (2.20), (2.22).

Введем матрицу

$$\mathbf{M}_1 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -t_{21} & 1 & 0 & \dots & 0 \\ -t_{31} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ -t_{N1} & 0 & 0 & \dots & 1 \end{pmatrix}.$$

Как не трудно проверить, справедливы равенства $\mathbf{A}^{(1)} = \mathbf{M}_1\mathbf{A}$, $\mathbf{b}^{(1)} = \mathbf{M}_1\mathbf{b}$, т.е. преобразование системы $\mathbf{Ax} = \mathbf{b}$ к виду (2.42) эквивалентно умножению левой и правой частей системы на матрицу $\mathbf{M}_1$.

Аналогично можно показать, что вычисления 2-го шага исключения приводят систему (2.42) к виду

$$\mathbf{A}^{(2)}\mathbf{x} = \mathbf{b}^{(2)},$$

где

$$\mathbf{A}^{(2)} = \mathbf{M}_2\mathbf{A}^{(1)}, \ \mathbf{b}^{(2)} = \mathbf{M}_2\mathbf{b}^{(1)};$$

$$\mathbf{A}^{(2)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2N}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3N}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a_{N3}^{(2)} & \dots & a_{NN}^{(2)} \end{pmatrix}, \quad \mathbf{b}^{(2)} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \dots \\ b_N^{(2)} \end{pmatrix},$$

$$\mathbf{M}_2 = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & -t_{32} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & -t_{N2} & 0 & \dots & 1 \end{pmatrix}.$$

После $(N - 1)$-го шага, завершающего прямой ход, система оказывается приведенной к виду

$$\mathbf{A}^{(N-1)}\mathbf{x} = \mathbf{b}^{(N-1)}, \tag{2.43}$$

с верхней треугольной матрицей $\mathbf{A}^{(N-1)}$. Здесь

$$\mathbf{A}^{(N-1)} = \mathbf{M}_{N-1}\mathbf{A}^{(N-2)},\ \mathbf{b}^{(N-1)} = \mathbf{M}_{N-1}\mathbf{b}^{(N-2)},$$

где

$$\mathbf{A}^{(N-1)} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2N}^{(1)} \\ 0 & 0 & a_{33}^{(2)} & \dots & a_{3N}^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{NN}^{(N-1)} \end{pmatrix},\ \mathbf{M}_{N-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & -t_{N,N-1} & 1 \end{pmatrix},$$

$$\mathrm{b}^{(N-1)} = \begin{pmatrix} b_1 \\ b_2^{(1)} \\ b_3^{(2)} \\ \dots \\ b_N^{(N-1)} \end{pmatrix}.$$

Заметим, что матрица $\mathbf{A}^{(N-1)}$ получена из матрицы $\mathbf{A}$ последовательным умножением на $\mathbf{M}_1$, $\mathbf{M}_2$, …, $\mathbf{M}_{N-1}$:

$$\mathbf{A}^{(N-1)} = \mathbf{M}_{N-1}\dots\mathbf{M}_2\mathbf{M}_1\mathbf{A}. \tag{2.44}$$

Аналогично,

$$\mathbf{b}^{(N-1)} = \mathbf{M}_{N-1}\dots\mathbf{M}_2\mathbf{M}_1\mathbf{b}. \tag{2.45}$$

Из равенства (2.44) вытекает следующее представление:

$$\mathbf{A} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\dots\mathbf{M}_{N-1}^{-1}\mathbf{A}^{(N-1)}. \tag{2.46}$$

Как легко проверить,

$$\mathbf{M}_1^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ t_{21} & 1 & 0 & \dots & 0 \\ t_{31} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ t_{N1} & 0 & 0 & \dots & 1 \end{pmatrix},\ \mathbf{M}_2^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & t_{32} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & t_{N2} & 0 & \dots & 1 \end{pmatrix},\ \dots,$$

$$\mathbf{M}_{N-1}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & t_{N,N-1} & 1 \end{pmatrix}.$$

Для этого достаточно перемножить матрицы $\mathbf{M}_k^{-1}$ и $\mathbf{M}_k$ ($k = 1, \ldots, N-1$), в результате чего получится единичная матрица.

Введем обозначения $\mathbf{U} = \mathbf{A}^{(N-1)}$, $\mathbf{L} = \mathbf{M}_1^{-1}\mathbf{M}_2^{-1}\ldots\mathbf{M}_{N-1}^{-1}$. Вычисляя матрицу **L**, убеждаемся в том, что она имеет следующий вид:

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ t_{21} & 1 & 0 & \ldots & 0 \\ t_{31} & t_{32} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ t_{N1} & t_{N2} & t_{N3} & \ldots & 1 \end{pmatrix}. \qquad (2.47)$$

Тогда равенство (2.46) в новых обозначениях примет вид

$$\mathbf{A} = \mathbf{LU}. \qquad (2.48)$$

Таким образом, получено LU-разложение исходной матрицы.

Очевидно, что решение СЛАУ с помощью LU-разложения – это другая схема реализации метода Гаусса. В отличие от рассмотренной ранее схемы единственного деления эту называют компактной схемой Гаусса.

В отличие от схемы единственного деления данная схема менее удобна для усовершенствования с целью уменьшения влияния вычислительных погрешностей путем выбора подходящих ведущих элементов. Достоинством же ее можно считать то, что LU-разложение матрицы **A** играет роль обратной матрицы, и может помещаться в памяти компьютера на место матрицы **A** и использоваться, например, при решении нескольких систем, имеющих одну и ту же матрицу коэффициентов и разные векторы свободных членов.

2.13. Обращение матриц с помощью LU-разложения

Для обращения матрицы **A** с помощью LU-разложения можно N-кратно использовать формулы (2.40) – (2.41) для получения столбцов матрицы $\mathbf{A}^{-1}$; при этом в качестве b_i в (2.40) должны фигурировать только 0 и 1: для нахождения первого столбца полагают $b_1 = 1$, $b_2 = 0$, $b_3 = 0$, … = $= b_N = 0$, для второго $b_1 = 0$, $b_2 = 1$, $b_3 = 0$, … $= b_N = 0$, и т.д. В результате после окончания N шагов на месте исходной матрицы **A** находится ее обратная матрица $\mathbf{A}^{-1}$.

2.14. Разложение симметричных матриц. Метод Холецкого (метод квадратных корней)

Пусть требуется решить систему линейных алгебраических уравнений

$$\mathbf{Ax} = \mathbf{b} \qquad (2.49)$$

с симметричной матрицей **A**. Системы уравнений такого типа часто встречаются в приложениях (например, в задачах оптимизации, при решении уравнений математической физики и др.). Для решения весьма часто применяется метод Холецкого[1]. В случае же если **A** еще и положительно определенная матрица, данный частный случай чаще называют методом квадратных корней.

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

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T. \tag{2.50}$$

В разложении (2.50) нижняя треугольная матрица

$$\mathbf{L} = \begin{pmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{N1} & l_{N2} & \dots & l_{NN} \end{pmatrix} \tag{2.51}$$

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 (2.50) получено, то решение системы (2.49) сводится к последовательному решению двух систем с треугольными матрицами:

$$\mathbf{L}\mathbf{y} = \mathbf{b},\ \mathbf{L}^T\mathbf{x} = \mathbf{y}. \tag{2.52}$$

Для решения систем (2.52) требуется выполнение примерно $2N^2$ арифметических операций.

Найдем элементы матрицы **L**. Для этого вычислим элементы матрицы $\mathbf{L}\mathbf{L}^T$ и приравняем их к соответствующим элементам матрицы **A**. В результате получим систему уравнений.

$$\begin{aligned}
&l_{11}^2 = a_{11};\\
&l_{i1}l_{11} = a_{i1},\ i = 2, 3, \dots, N;\\
&l_{21}^2 + l_{22}^2 = a_{22};\\
&l_{i1}l_{21} + l_{i2}l_{22} = a_{i2},\ i = 3, 4, \dots, N;\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&l_{k1}^2 + l_{k2}^2 + \dots + l_{kk}^2 = a_{kk};\\
&l_{i1}l_{k1} + l_{i2}l_{k2} + \dots + l_{ik}l_{kk} = a_{ik},\ i = k+1, \dots, N;\\
&\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\\
&l_{N1}^2 + l_{N2}^2 + \dots + l_{NN}^2 = a_{NN}.
\end{aligned} \tag{2.53}$$

[1] Андре-Луи Холецкий (1875–1918) – французский математик и геодезист, лейтенант артиллерии. Погиб во время Первой мировой войны.

Решая систему (2.53), последовательно находим

$$
\begin{aligned}
& l_{11} = \sqrt{a_{11}}; \\
& l_{i1} = a_{i1} / l_{11},\ i = 2, 3, \ldots, N; \\
& l_{22} = \sqrt{a_{22} - l_{21}^2}; \\
& l_{i2} = (a_{i2} - l_{i1} l_{21}) / l_{22},\ i = 3, 4, \ldots, N; \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& l_{kk} = \sqrt{a_{kk} - l_{k1}^2 - l_{k2}^2 - \ldots - l_{k,k-1}^2}; \\
& l_{ik} = (a_{ik} - l_{i1} l_{k1} - l_{i2} l_{k2} - \ldots - l_{i,\,k-1} l_{k,\,k-1}) / l_{kk},\ i = k + 1, \ldots, N; \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
& l_{NN} = \sqrt{a_{NN} - l_{N1}^2 - l_{N2}^2 - \ldots - l_{N,N-1}^2}.
\end{aligned}
\qquad (2.54)
$$

Заметим, что для вычисления диагональных элементов используется операция извлечения квадратно корня. Поэтому метод Холецкого называют еще и методом квадратных корней. Доказано, что положительность соответствующих подкоренных выражений является следствием положительной определенности матрицы **A**.

Метод обладает рядом ценных качеств, которые позволяют предпочесть его методу Гаусса, если требуется решить СЛАУ с симметричной и положительно определенной матрицей.

Как нетрудно подсчитать, число операций, выполняемых в ходе выполнения (2.50) по формулам (2.54), равно примерно $N^3 / 3$. Учитывая, что для решения систем (2.52) требуется примерно $2N^2$ арифметических операций, убеждаемся, что при больших N метод Холецкого 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} ($i \geq j$), расположенные на главной диагонали и под ней. В формулах (2.54) 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 **A**. Применение для решения системы (2.49) метода Гаусса потребовало бы использования примерно вдвое большего объема памяти. Безусловным достоинством метода Холецкого является также его гарантированная устойчивость.

Пример 2.14. Используя метод Холецкого, найдем решение системы уравнений с симметричной положительно определенной матрицей:

$$\begin{aligned} 6.25x_1 &- x_2 + 0.5x_3 = 7.5, \\ -x_1 &+ 5x_2 + 2.12x_3 = -8.68, \\ 0.5x_1 &+ 2.12x_2 + 3.6x_3 = -0.24. \end{aligned}$$

По формулам (2.54) последовательно находим:

$l_{11} = \sqrt{a_{11}} = \sqrt{6.25} = 2.5, \quad l_{21} = a_{21} / l_{11} = -1 / 2.5 = -0.4;$

$l_{31} = a_{31} / l_{11} = 0.5 / 2.5 = 0.2, \; l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{5 - 0.16} = 2.2;$

$l_{32} = (a_{32} - l_{31}l_{21}) / l_{22} = (2.12 - 0.2(-0.4)) / 2.2 = 1;$

$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{3.6 - 0.2^2 - 1^2} = 1.6.$

Следовательно, матрица **L** такова:

$$\mathbf{L} = \begin{pmatrix} 2.5 & 0 & 0 \\ -0.4 & 2.2 & 0 \\ 0.2 & 1 & 1.6 \end{pmatrix}.$$

Система **Ly** = **b** имеет вид

$$\begin{aligned} 2.5y_1 &= 7.5, \\ -0.4y_1 &+ 2.2y_2 = -8.68, \\ 0.2y_1 &+ y_2 + 1.6y_3 = -0.24. \end{aligned}$$

Решая ее, получим $y_1 = 3$, $y_2 = -3.4$, $y_3 = 1.6$.

Далее из системы $\mathbf{L}^T\mathbf{x} = \mathbf{y}$, которая имеет вид

$$\begin{aligned} 2.5x_1 &- 0.4x_2 + 0.2x_3 = 3, \\ 2.2x_2 &+ x_3 = -3.4, \\ 1.6x_3 &= 1.6, \end{aligned}$$

находим решение $x_1 = 0.8$, $x_2 = -2$, $x_3 = 1$.

2.15. Метод прогонки

Рассмотрим метод прогонки[1] – простой и эффективный алгоритм решения СЛАУ с трехдиагональными матрицами.

[1] Метод прогонки был предложен в начале 50-х гг. независимо несколькими авторами, в том числе российскими учеными И.М. Гельфандом, О.В. Локуциевским, В.С. Владимировым, А.С. Кронродом.

$$
\begin{array}{ccccccccccc}
b_1x_1 & + & c_1x_2 & & & & & & & = & d_1, \\
a_2x_1 & + & b_2x_2 & + & c_2x_3 & & & & & = & d_2, \\
 & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
 & a_ix_{i-1} & + & b_ix_i & + & c_ix_{i+1} & & & & = & d_i, \\
 & & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \\
 & & & a_{N-1}x_{N-2} & + & b_{N-1}x_{N-1} & + & c_{N-1}x_N & = & d_{N-1}, & \\
 & & & & & a_Nx_{N-1} & + & b_Nx_N & = & d_N. &
\end{array}
\qquad (2.55)
$$

Системы такого вида часто возникают при решении различных задач математической физики, а также при решении других вычислительных задач (например, приближения функций сплайнами).

Преобразуем первое уравнение (2.55) к виду

$$x_1 = \alpha_1 x_2 + \beta_1, \qquad (2.56)$$

где $\alpha_1 = -c_1 / b_1$, $\beta_1 = d_1 / b_1$.

Подставим выражение для x_1 во второе уравнение системы:

$$a_2(\alpha_1 x_2 + \beta_1) + b_2x_2 + c_2x_3 = d_2.$$

Преобразуем это уравнение к виду

$$x_2 = \alpha_2 x_3 + \beta_2, \qquad (2.57)$$

где $\alpha_2 = -c_2 / (b_2 + a_2\alpha_1)$, $\beta_2 = (d_2 - a_2\beta_1) / (b_2 + a_2\alpha_1)$.

Выражение (2.57) подставляем в третье уравнение системы и т.д.

На i-м шаге этого процесса ($1 < i < N$) i-е уравнение системы преобразуется к виду

$$x_i = \alpha_i x_{i+1} + \beta_i, \qquad (2.58)$$

где $\alpha_i = -c_i / (b_i + a_i\alpha_{i-1})$, $\beta_i = (d_i - a_i\beta_{i-1}) / (b_i + a_i\alpha_{i-1})$.

На N-м шаге подстановка в последнее уравнение выражения $x_{N-1} = \alpha_N x_{N+1} + \beta_{N-1}$ дает:

$$a_N(\alpha_{N-1}x_N + \beta_{N-1}) + b_Nx_N = d_N.$$

Откуда можно определить значение x_N:

$$x_N = \beta_N = (d_N - a_N\beta_{N-1}) / (b_N + a_N\alpha_{N-1}).$$

Значения остальных неизвестных x_i для $i = N-1, N-2, \ldots, 1$ теперь легко вычисляются по формуле (2.58).

Сделанные преобразования позволяют организовать вычисления метода прогонки в два этапа.

Прямой ход метода прогонки (прямая прогонка) состоит в вычислении прогоночных коэффициентов α_i ($1 \le i < N$) и β_i ($1 \le i < N$). При $i = 1$ коэффициенты вычисляют по формулам

$$\alpha_1 = -c_1 / \gamma_1, \beta_1 = d_1 / \gamma_1, \gamma_1 = b_1, \qquad (2.59)$$

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

$$\alpha_i = -c_i / \gamma_i,\ \beta_i = (d_i - a_i\beta_{i-1}) / \gamma_i,\ \gamma_i = b_i + a_i\alpha_{i-1}. \quad (2.60)$$

При $i = N$ прямая прогонка завершается вычислениями

$$\beta_N = (d_N - a_N\beta_{N-1}) / \gamma_N,\ \gamma_N = b_N + a_N\alpha_{N-1}. \quad (2.61)$$

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

$$x_i = \alpha_i x_{i+1} + \beta_i,\ i = N-1, N-2, \ldots, 1. \quad (2.62)$$

Вычисления ведут в порядке убывания значений i от $N - 1$ до 1.

Пример 2.15. Используя метод прогонки, решим систему

$$\begin{aligned}
5x_1 &- x_2 &&&&&&= 2.0,\\
2x_1 &+ 4.6x_2 &- x_3 &&&&&= 3.3,\\
&\quad 2x_2 &+ 3.6x_3 &- 0.8x_4 &&&&= 2.6,\\
&&\quad 3x_3 &+ 4.4x_4 &&&&= 7.2.
\end{aligned}$$

Прямой ход. Согласно формулам (2.59) – (2.61) получаем:

$\gamma_1 = b_1 = 5$, $\alpha_1 = -c_1 / \gamma_1 = 1 / 5 = 0.2$, $\beta_1 = d_1 / \gamma_1 = 2.0 / 5 = 0.4$;

$\gamma_2 = b_2 + a_2\alpha_1 = 4.6 + 2\cdot 0.2 = 5$, $\alpha_2 = -c_2 / \gamma_2 = 1 / 5 = 0.2$;

$\beta_2 = (d_2 - a_2\beta_1) / \gamma_2 = (3.3 - 2\cdot 0.4) / 5 = 0.5$;

$\gamma_3 = b_3 + a_3\alpha_2 = 3.6 + 2\cdot 0.2 = 4$, $\alpha_3 = -c_3 / \gamma_3 = 0.8 / 4 = 0.2$;

$\beta_3 = (d_3 - a_3\beta_2) / \gamma_3 = (2.6 - 2\cdot 0.5) / 4 = 0.4$;

$\gamma_4 = b_4 + a_4\alpha_3 = 4.4 + 3\cdot 0.2 = 5$;

$\beta_4 = (d_4 - a_4\beta_3) / \gamma_4 = (7.2 - 3\cdot 0.4) / 5 = 1.2$.

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 формулам (2.59) – (2.62) требуется примерно $8N$ арифметических операций, тогда как в методе Гаусса это число составляет примерно $(2/3)N^3$. Важно и то, что трехдиагональная структура матрицы системы позволяет использовать для ее хранения $3N - 2$ 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 плотной (заполненной) матрицей.

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

$$\mathbf{A} = \begin{pmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & a_N & b_N \end{pmatrix}$$

представляется в виде произведения двух двудиагональных матриц:

$$\mathbf{A} = \mathbf{L}\mathbf{U}. \qquad (2.63)$$

Здесь

$$\mathbf{L} = \begin{pmatrix} \gamma_1 & 0 & 0 & \dots & 0 & 0 \\ a_2 & \gamma_2 & 0 & \dots & 0 & 0 \\ 0 & a_3 & \gamma_3 & \dots & 0 & 0 \\ & & & & & \\ 0 & 0 & 0 & \dots & a_N & \gamma_N \end{pmatrix}, \quad \mathbf{U} = \begin{pmatrix} 1 & -\alpha_1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -\alpha_2 & \dots & 0 & 0 \\ & & & & & \\ 0 & 0 & 0 & \dots & 1 & -\alpha_{N-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{pmatrix}.$$

Так как для определения **L** и **U** нет необходимости вычислять коэффициенты β_i, то общее число операций на получение разложения (2.63) составляет примерно $3N$.

Разложение (2.63) можно использовать для решения систем со многими правыми частями. Если нужно решить p систем с матрицей **A**, то общее число операций составит примерно $3N + 5Np$.

К сожалению, при обращении матрицы **A** теряется ее трехдиагональная структура. Обратная матрица является заполненной, однако для ее вычисления с помощью разложения (2.63) требуется примерно лишь $2.5N$ арифметических операций.

Наряду с изложенным выше «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 (2.55) структуре, разработаны методы циклической прогонки, матричной прогонки и др. С указанными вариантами метода прогонки можно подробно ознакомиться в [9, 10].

2.16. Метод исключения Жордана (Гаусса–Жордана)

Еще одной из модификаций описанного выше метода Гаусса является подход, предложенный Жорданом[1], позволяющий получить решения с помощью только прямого хода. Различие этих двух методов заключается в том, что при реализации последнего элементы матрицы обнуляются как под, так и над главной диагональю. В результате данного подхода вектор решения находится на месте вектора свободных членов, а на месте исходной матрицы находится единичная.

Рассмотрим подробнее данный метод, без выбора ведущего элемента.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1N} & a_{2N} & \cdots & a_{NN} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_N \end{pmatrix}$$

Пусть a_{11} ведущий элемент. На первом этапе первая строка подвергается нормировке ведущим элементом, а затем аналогично методу Гаусса происходит обнуление поддиагональных элементов. В результате первого этапа получаем эквивалентную систему

$$\begin{pmatrix} 1 & a_{12}^{(1)} = a_{12}/a_{11} & \cdots & a_{1N}^{(1)} = a_{1N}/a_{11} \\ 0 & a_{22}^{(1)} & \cdots & a_{2N}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & a_{N2}^{(1)} & \cdots & a_{NN}^{(1)} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1^{(1)} = b_1/a_{11} \\ b_2^{(1)} \\ \cdots \\ b_N^{(1)} \end{pmatrix}.$$

На втором этапе после нормировки второй строки на $a_{22}^{(1)}$, аналогично методу Гаусса обнуляют поддиагональные элементы, а также и наддиагональный. Таким образом, после второго шага система имеет вид

$$\begin{pmatrix} 1 & 0 & \cdots & a_{1N}^{(2)} \\ 0 & 1 & \cdots & a_{2N}^{(2)} = a_{2N}^{(1)}/a_{22}^{(1)} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{NN}^{(2)} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1^{(2)} \\ b_2^{(2)} = b_2^{(1)}/a_{1N}^{(2)} \\ \cdots \\ b_N^{(2)} \end{pmatrix}.$$

Продолжая подобным образом, на месте исходной матрицы получим единичную, а на месте вектора свободных членов вектор решения.

Далее приведен алгоритм метода Гаусса–Жордана.

---

[1] Камиль Мари Эдмон Жордан (1838–1922) – французский математик, внесший значительный вклад в алгебру, теорию чисел, теорию функций, геометрию, топологию и дифференциальные уравнения.

Алгоритм метода Гаусса–Жордана

Для $k = 1, 2, \ldots, N$
  $b_k = b_k / a_{kk}$
  Для $j = N, N-1, \ldots, k$
    $a_{kj} = a_{kj} / a_{kk}$
  Увеличить j
  Для $i = 1, 2, \ldots k-1, k+1, \ldots, N$
    $b_i = b_i - a_{ik}\, b_k$
  Увеличить i
  Для $j = N, N-1, \ldots, k$
    Для $i = 1, 2, \ldots k-1, k+1, \ldots, N$
      $a_{ij} = a_{ij} - a_{ik}\, a_{kj}$
    Увеличить i
  Увеличить i
Увеличить k

Для повышения точности используют модификацию с частичным или полным выбором ведущего элемента. Для введения частичного упорядочивания необходимо, как и в алгоритме метода Гаусса, сделать вставку, приведенную в разделе 2.8.

Пример 2.16. Решим систему с помощью метода Гаусса–Жордана

$$\begin{aligned} 80x_1 &- 20x_2 - 20x_3 = 20, \\ -20x_1 &+ 40x_2 - 20x_3 = 20, \\ -20x_1 &- 20x_2 + 130x_3 = 20. \end{aligned}$$

После нормировки первой строки с помощью элемента a_{11}, система примет вид

$$\begin{aligned} 1x_1 &- (1/4)x_2 - (1/4)x_3 = 1/4, \\ -20x_1 &+ 40x_2 - 20x_3 = 20, \\ -20x_1 &- 20x_2 + 130x_3 = 20. \end{aligned}$$

Умножим первое уравнение на (–20), вычтем его из второго и третьего уравнений и получим

$$\begin{aligned} 1x_1 &- (1/4)x_2 - (1/4)x_3 = 1/4, \\ &35x_2 - 25x_3 = 25, \\ -&25x_2 + 125x_3 = 25. \end{aligned}$$

После нормировки второго уравнения элементом $a_{22}^{(1)}$ система примет вид

$$
\begin{aligned}
1x_1 &- (1/4)x_2 &- (1/4)x_3 &= 1/4\,,\\
&1x_2 &- (5/7)x_3 &= 5/7\,,\\
-\ &25x_2 &+\ 125x_3 &= 25\,.
\end{aligned}
$$

Умножив второе уравнение на (–1/4) вычтем его из первого уравнения, а умножив на (–25), вычтем его из третьего уравнения. После преобразований получим систему

$$
\begin{aligned}
1x_1 &- (3/7)x_3 &= 3/7\,,\\
1x_2 &- (5/7)x_3 &= 5/7\,,\\
+\ &750/7x_3 &= 300/7\,.
\end{aligned}
$$

Пронормировав третье уравнение с помощью $a_{33}^{(2)}$, умножим его на (–3/7) и вычтем из первого, далее умножим на (–5/7) и вычтем из второго уравнения. После преобразований система примет вид

$$
\begin{aligned}
1x_1 &= 0.6\,,\\
1x_2 &= 1.0\,,\\
1x_3 &= 0.4\,.
\end{aligned}
$$

Таким образом, полученное решение: $x_1 = 0.6$, $x_2 = 1.0$, $x_3 = 0.4$.

2.17. QR-разложение матрицы

Методы Гаусса и Гаусса–Жордана не являются единственными методами исключения, используемыми для решения систем линейных уравнений и приведения матриц к треугольному виду. 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**[1] на верхнюю треугольную матрицу **R**:

$$\mathbf{A} = \mathbf{QR}. \qquad (2.64)$$

Представление (2.64) – это QR-разложение[2] матриц на множители.

---

[1] Вещественная матрица **Q** называется ортогональной, если для нее выполнено условие $\mathbf{Q}^T = \mathbf{Q}^{-1}$, что эквивалентно равенствам $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{E}$. 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: $\|\mathbf{Qx}\|_2 = \|\mathbf{x}\|_2$ для всех $\mathbf{x} \in R^N$.

[2] Это разложение было предложено почти одновременно российским математиком В.Н. Кублаковской (1961) и англичанином Дж. Фрэнсисом (1962).

2.17.1. *Метод вращений*

Опишем прямой ход метода. На 1-м шаге неизвестное x_1 исключают из всех уравнений, кроме первого. Для исключения x_1 из второго уравнения вычисляют числа

$$c_{12} = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}, \quad s_{12} = \frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}, \tag{2.65}$$

обладающими следующими свойствами:

$$c_{12}^2 + s_{12}^2 = 1, \quad -s_{12}a_{11} + c_{12}a_{21} = 0. \tag{2.66}$$

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 коэффициентами $-s_{12}$ и c_{12}. В результате получают систему

$$\begin{array}{ccccccccccc}
a_{11}^{(1)}x_1 & + & a_{12}^{(1)}x_2 & + & a_{13}^{(1)}x_3 & + & \ldots & + & a_{1N}^{(1)}x_N & = & b_1^{(1)}, \\
 & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & \ldots & + & a_{2N}^{(1)}x_N & = & b_2^{(1)}, \\
a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & + & \ldots & + & a_{3N}x_N & = & b_3, \\
\ldots\ldots & & \ldots\ldots & & \ldots\ldots & & \ldots & & \ldots\ldots & \ldots & \ldots \\
a_{N1}x_1 & + & a_{N2}x_2 & + & a_{N3}x_3 & + & \ldots & + & a_{NN}x_N & = & b_N,
\end{array} \tag{2.67}$$

в которой

$$\begin{gathered}
a_{1j}^{(1)} = c_{12}a_{1j} + s_{12}a_{2j}, \\
a_{2j}^{(1)} = -s_{12}a_{1j} + c_{12}a_{2j}, \quad (1 \le j \le N); \\
b_1^{(1)} = c_{12}b_1 + s_{12}b_2, \quad b_2^{(1)} = -s_{12}b_1 + c_{12}b_2.
\end{gathered} \tag{2.68}$$

Заметим, что $a_{21}^{(1)} = -s_{12}a_{11} + c_{12}a_{21} = 0$ в силу специального выбора чисел c_{12} и s_{12} (2.66). Естественно, что если $a_{21} = 0$, исходная система уже имеет вид (2.67) и в исключении неизвестного x_1 из второго уравнения нет необходимости. В этом случае полагают $c_{12} = 1$ и $s_{12} = 0$.

Таким образом, преобразование исходной системы (2.19) к виду (2.67) эквивалентно умножению слева матрицы **A** и правой части **b** на матрицу

$$\mathbf{G}_{12} = \begin{pmatrix}
c_{12} & s_{12} & 0 & 0 & \ldots & 0 \\
-s_{12} & c_{12} & 0 & 0 & \ldots & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
0 & 0 & 0 & 0 & \ldots & 1
\end{pmatrix}.$$

Для исключения неизвестного x_1 из третьего уравнения вычисляют

$$c_{13} = \frac{a_{11}^{(1)}}{\sqrt{\left(a_{11}^{(1)}\right)^2 + a_{31}^2}}, \quad s_{13} = \frac{a_{31}}{\sqrt{\left(a_{11}^{(1)}\right)^2 + a_{31}^2}}, \tag{2.69}$$

такие, что $c_{13}^2 + s_{13}^2 = 1$, $-s_{13}a_{11}^{(1)} + c_{13}a_{31} = 0$. Затем первое уравнение системы (2.67) заменяют линейной комбинацией первого и третьего уравнений с коэффициентами c_{13} и s_{13}, а третье уравнение – аналогичной комбинацией с коэффициентами $-s_{13}$ и c_{13}. Это преобразование системы эквивалентно умножению слева на матрицу

$$\mathbf{G}_{13} = \begin{pmatrix} c_{13} & 0 & s_{13} & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 \\ -s_{13} & 0 & c_{13} & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 1 \end{pmatrix}$$

и приводит к тому, что коэффициент при x_1 в преобразованном третьем уравнении обращается в нуль.

Матрицы $\mathbf{G}_{ij}$ называются матрицами плоских вращений. Данный метод QR-разложения матрицы также часто называют преобразованиями Гивенса[1].

Пример 2.17. Используя метод вращений, решим на 6-разрядном десятичном компьютере следующую систему линейных алгебраических уравнений

$$\begin{aligned} 2x_1 &- 9x_2 &+ 5x_3 &= -4, \\ 1.2x_1 &- 5.3999x_2 &+ 6x_3 &= 0.6001, \\ x_1 &- x_2 &- 7.5x_3 &= -8.5. \end{aligned}$$

Прямой ход.1-й шаг. Исключим x_1 из второго уравнения. Для этого вычислим c_{12} и s_{12} по формулам (2.69):

$$c_{12} = \frac{2}{\sqrt{2^2 + 1.2^2}} \approx 0.857493, \quad s_{12} = \frac{1.2}{\sqrt{2^2 + 1.2^2}} \approx 0.514495.$$

Преобразуя коэффициенты первого и второго уравнения по формулам (2.68), приходим к системе

[1] Гивенс Джеймс Уоллас (1910–1993) – американский математик.

$$\begin{aligned} 2.33238x_1 &- 10.4957x_2 &+ 7.37444x_3 &= -3.12122,\\ &7.85493\cdot 10^{-5}x_2 &+ 2.57248x_3 &= 2.57256,\\ x_1 &- x_2 &- 7.5x_3 &= -8.5. \end{aligned}$$

Далее вычислим коэффициенты c_{13} и s_{13} по формулам (2.69):

$$c_{13} = \frac{2.33238}{\sqrt{2.33238^2 + 1^2}} \approx 0.919087, \quad s_{13} = \frac{1}{\sqrt{2.33238^2 + 1^2}} \approx 0.394055.$$

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

$$\begin{aligned} 2.53772x_1 &- 10.0405x_2 &+ 3.82234x_3 &= -6.21814,\\ &7.85493\cdot 10^{-5}x_2 &+ 2.57248x_3 &= 2.57256,\\ &3.21680x_2 &- 9.79909x_3 &= -6.58231. \end{aligned}$$

2-й шаг. В полученной системе имеем $a_{22}^{(1)} = 7.85493\cdot 10^{-5}$, $a_{32}^{(1)} = 3.21680$. Поэтому

$$c_{23} = \frac{a_{11}^{(1)}}{\sqrt{\left(a_{22}^{(1)}\right)^2 + a_{32}^2}} \approx 2.44185\cdot 10^{-5}, \quad s_{13} \approx 1.00000\,.$$

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

$$\begin{aligned} 2.53772x_1 &- 10.0405x_2 &+ 3.82234x_3 &= -6.21814,\\ &3.21680x_2 &+ 9.79903x_3 &= -6.58225,\\ & &- 2.57272x_3 &= -2.57272. \end{aligned}$$

Обратный ход дает последовательно значения $x_3 = 1$, $x_2 = 0.999994$, $x_1 = -1.58579\cdot 10^{-5}$.

2.17.2. *Метод отражений*

Матрицами Хаусхолдера[1] (или отражений) называются квадратные матрицы вида

$$\mathbf{V} = \mathbf{E} - 2\mathbf{w}\mathbf{w}^T,$$

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

[1] Хаусхолдер Олстон (1904–1993) – американский математик. Занимался вопросами вариационного исчисления, математической биологии, численного анализа. Как и Гивенс, в свое время был президентом организации SIAM (общество промышленной и прикладной математики).

Матрица Хаусхолдера симметрична и ортогональна. Действительно,

$$\mathbf{V}^T = (\mathbf{E} - 2\mathbf{w}\mathbf{w}^T)^T = \mathbf{E}^T - 2(\mathbf{w}^T)^T\mathbf{w}^T = \mathbf{E} - 2\mathbf{w}\mathbf{w}^T = \mathbf{V};$$

$$\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V} = (\mathbf{E} - 2\mathbf{w}\mathbf{w}^T)(\mathbf{E} - 2\mathbf{w}\mathbf{w}^T) = \mathbf{E} - 4\mathbf{w}\mathbf{w}^T + 4\mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T = \mathbf{E}.$$

Здесь учтено, что $\mathbf{w}\mathbf{w}^T = \| \mathbf{w} \|_2^2 = 1$. Умножение на матрицу $\mathbf{V}$ называют преобразованием Хаусхолдера (или отражением).

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 $(4/3)N^3$ арифметических операций, т.е. в полтора раза быстрее. Детальное изложение самого метода можно найти, например в [11].

Поясним, как получается QR-разложение матрицы $\mathbf{A}$ с помощью преобразований Хаусхолдера. Пусть $\mathbf{a} = (a_1, a_2, \ldots, a_N)^T$ – произвольный вектор, у которого одна из координат $a_2, \ldots, a_N$ отлична от нуля. Покажем, что вектор $\mathbf{w}$ в преобразовании Хаусхолдера может быть выбран так, чтобы обратились в нуль все координаты вектора $\mathbf{V}\mathbf{a}$, кроме первой: $\mathbf{V}\mathbf{a} = c\mathbf{e}_1 = c\,(1,\ 0,\ \ldots,\ 0)^T$. Поскольку ортогональное преобразование не меняет евклидову норму вектора, то $c = \|\mathbf{a}\|_2$ и искомое преобразование таково, что

$$\mathbf{V}\mathbf{a} = (\mathbf{E} - 2\mathbf{w}\mathbf{w}^T)\mathbf{a} = \mathbf{a} - 2(\mathbf{w}, \mathbf{a})\mathbf{w} = \mathbf{a} - \alpha\mathbf{w} = \|\mathbf{a}\|_2\mathbf{e}_1,$$

где $\alpha = 2(\mathbf{w}, \mathbf{a})$. Таким образом, вектор $\mathbf{w}$ следует выбрать так, чтобы

$$\alpha w_1 - a_1 = \pm \|\mathbf{a}\|_2,\ \alpha w_1 = a_2,\ \ldots,\ \alpha w_N = a_N,$$

это эквивалентно равенству $\mathbf{w} = \alpha^{-1}(\mathbf{a} \pm \|\mathbf{a}\|_2\mathbf{e}_1)$. Вспоминая, что $\alpha = 2(\mathbf{w}, \mathbf{a}) = 2(w_1a_1 + w_2a_2 + \ \ldots + w_Na_N)$, имеем

$$\alpha^2 = 2\,(a_1 \pm \| \mathbf{a} \|_2)a_1 + 2\,(a_2^2 + \ldots + a_N^2) = \| \mathbf{a} \pm \| \mathbf{a} \|_2\mathbf{e}_1 \|_2^2 .$$

Таким образом, $\mathbf{w} = \dfrac{\boldsymbol{\upsilon}}{\| \boldsymbol{\upsilon} \|_2}$, где $\boldsymbol{\upsilon} = \mathbf{a} \pm \|\mathbf{a}\|_2\mathbf{e}_1$.

Взяв $\mathbf{a} = \mathbf{a}_1$, где $\mathbf{a}_1$ – первый из столбцов матрицы $\mathbf{A}$, и положив $\mathbf{P}_1 = \mathbf{V}$, получим

$$\mathbf{A}^{(1)} = \mathbf{P}_1\mathbf{A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \ldots & a_{1N}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \ldots & a_{2N}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \ldots & a_{3N}^{(1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & a_{N2}^{(1)} & a_{N3}^{(1)} & \ldots & a_{NN}^{(1)} \end{pmatrix}.$$

Далее, взяв вектор $\mathbf{a}_2 = (a_{22}^{(1)}, \ldots, a_{N2}^{(1)})^T \in R^{N-1}$, с помощью преобразования Хаусхолдера $\mathbf{V}_{N-1}$ в пространстве $(N-1)$-мерных векторов можно обнулить все координаты вектора $\mathbf{V}_{N-1}\mathbf{a}_2$ кроме первой. Положив

$$\mathbf{P}_2 = \begin{pmatrix} 1 & 0 \\ 0 & \mathbf{V}_{N-1} \end{pmatrix},$$

получим:

$$\mathbf{A}^{(2)} = \mathbf{P}_2\mathbf{A}^{(1)} = \mathbf{P}_2\mathbf{P}_1\mathbf{A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \ldots & a_{1N}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \ldots & a_{2N}^{(2)} \\ 0 & 0 & a_{33}^{(2)} & \ldots & a_{3N}^{(2)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & a_{N3}^{(2)} & \ldots & a_{NN}^{(2)} \end{pmatrix}.$$

Заметим, что первый столбец и первая строка матрицы при этом преобразовании не меняется.

Выполнив $N-1$ шаг этого метода, приходим к верхней треугольной матрице

$$\mathbf{A}^{(N-1)} = \mathbf{P}_{N-1}\ldots\mathbf{P}_2\mathbf{P}_1\mathbf{A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \ldots & a_{1N}^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \ldots & a_{2N}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \ldots & a_{3N}^{(3)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & a_{NN}^{(N-1)} \end{pmatrix}.$$

Поскольку матрицы $\mathbf{P}_1, \ldots, \mathbf{P}_{N-1}$ симметричны и ортогональны, получено разложение

$$\mathbf{A} = \mathbf{QR},$$

где $\mathbf{Q} = \mathbf{P}_1\mathbf{P}_2 \ldots\mathbf{P}_{N-1}$ – ортогональная, а матрица $\mathbf{R} = \mathbf{A}^{(N-1)}$ – верхняя треугольная.

Пример 2.18. Применим метод отражений для решения системы уравнений из примера 2.17.

$$\begin{aligned} 2x_1 &- 9x_2 &+ 5x_3 &= -4, \\ 1.2x_1 &- 5.3999x_2 &+ 6x_3 &= 0.6001, \\ x_1 &- x_2 &- 7.5x_3 &= -8.5. \end{aligned}$$

Прямой ход. Возьмем первый столбец матрицы $\mathbf{a}_1 = (2, 1.2, 1)^T$, положим $\boldsymbol{\upsilon} = \mathbf{a}_1 + \|\mathbf{a}_1\|_2\mathbf{e}_1 = (2 + \sqrt{6.44}, 1.2, 1)^T \approx (4.53772, 1.2, 1)^T$ и выберем

$$\mathbf{w} = \boldsymbol{\upsilon} / \|\boldsymbol{\upsilon}\|_2 \approx (0.945545, 0.250049, 0.208375)^T.$$

Построим матрицу Хаусхолдера

$$\mathbf{V} = \mathbf{E} - 2\mathbf{w}\mathbf{w}^T =$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 0.945545 \\ 0.250049 \\ 0.208375 \end{pmatrix} (0.945545 \quad 0.250049 \quad 0.208375) \approx$$

$$\approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 0.894055 & 0.236433 & 0.197028 \\ 0.236433 & 0.0625245 & 0.0521040 \\ 0.197028 & 0.0521040 & 0.0434201 \end{pmatrix} \approx$$

$$\approx \begin{pmatrix} -0.788110 & -0.472866 & -0.394056 \\ -0.472866 & 0.874951 & -0.104208 \\ -0.394056 & -0.104208 & 0.913160 \end{pmatrix}.$$

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

$$\begin{aligned} -2.53772x_1 + 10.0405x_2 + 3.82234x_3 &= 6.21814, \\ -0.364646x_2 + 3.66694x_3 &= 3.30229, \\ 3.19606x_2 - 9.44423x_3 &= -6.24817. \end{aligned}$$

Возьмем теперь $\mathbf{a}_2 = (-0.364646,\ 3.19606)^T$ и посмотрим двумерное преобразование Хаусхолдера. Здесь $\boldsymbol{\upsilon} = \mathbf{a}_2 + \|\mathbf{a}_2\|_2(1, 0)^T = (-0.364646 + +\sqrt{0.364646^2 + 3.19606^2},\ 3.19606\ )^T \approx (2.85215,\ 3.19606\ )^T$, $\mathbf{w} = \boldsymbol{\upsilon} / \|\boldsymbol{\upsilon}\|_2 \approx \approx (0.665824,\ 0.746109)^T$. Двумерная матрица Хаусхолдера имеет вид

$$\mathbf{V}_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - 2 \begin{pmatrix} 0.665824 \\ 0.746109 \end{pmatrix} (0.665824,\ 0.746109) \approx$$

$$\approx \begin{pmatrix} 0.113357 & -0.993555 \\ -0.993555 & -0.113357 \end{pmatrix}.$$

Умножив обе части системы на матрицу

$$\mathbf{P}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.113357 & -0.993555 \\ 0 & -0.993555 & -0.113357 \end{pmatrix},$$

получим

$$\begin{aligned} -2.53772x_1 + 10.0405x_2 + 3.82234x_3 &= 6.21814, \\ -3.21680x_2 + 9.79904x_3 &= 6.58224, \\ - 2.57274x_3 &= -2.57273. \end{aligned}$$

Обратный ход последовательно дает значения $x_3 \approx 0.999996$, $x_2 \approx 0.999988$, $x_1 \approx -3.35721 \cdot 10^{-5}$.

Алгоритмы ортогонального разложения отличает устойчивость и повышенная точность вычислений. Единственное, что требует доработки по сравнению с алгоритмами LU-разложения, это вопрос о рациональном хранении результатов разложения. Методы QR-разложения имеют аналогичные преимущества перед методами Гаусса и Гаусса–Жордана, что и методы LU-разложения при многократном решении систем уравнений с изменяющейся правой частью.

2.18. Итерационное уточнение

В большинстве случаев метод Гаусса с выбором главного элемента позволяет получить приближенное решение с довольно высокой точностью. Однако иногда возникает необходимость найти решение с большей точностью. Существует метод, позволяющий найти приближенное решение с относительной точностью, сравнимой с ε_M, если только число обусловленности не слишком велико. Этот метод, называемый итерационным уточнением, требует увеличения (примерно 25%) машинного времени по сравнению с затратами на получение решения методом Гаусса.

Пусть $\mathbf{x}^{(0)}$ – найденное на компьютере приближенное решение системы $\mathbf{Ax} = \mathbf{b}$. Напомним (см. 2.1), что невязка $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{Ax}^{(0)}$ и погрешность $\mathbf{e}^{(0)} = \mathbf{x} - \mathbf{x}^{(0)}$ связаны равенством

$$\mathbf{Ae}^{(0)} = \mathbf{r}^{(0)}. \qquad (2.70)$$

Если бы удалось найти $\mathbf{e}^{(0)}$ как точное решение системы (2.70), то вектор $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \mathbf{e}^{(0)}$ дал бы точное решение системы $\mathbf{Ax} = \mathbf{b}$. Однако в действительности вычисленное на компьютере значение $\mathbf{x}^{(1)}$ неизбежно будет содержать погрешность. Тем не менее, можно ожидать, что $\mathbf{x}^{(1)}$ окажется лучшим приближением, чем $\mathbf{x}^{(0)}$. Используя приближение $\mathbf{x}^{(1)}$, аналогичным образом можно найти приближение $\mathbf{x}^{(2)}$.

Опишем более подробно очередной $(k + 1)$-й шаг метода.

1. Вычисляют $\mathbf{r}^{(k)} \approx \mathbf{b} - \mathbf{Ax}^{(k)}$. Исключительно важно, чтобы вычисление $\mathbf{r}^{(k)}$ производилось с повышенной точностью. Дело в том, что $\mathbf{b} \approx \mathbf{Ax}^{(k)}$ и поэтому при вычислении невязки неизбежно вычитание близких чисел, а, следовательно, потеря большого числа значащих цифр. Одна из возможностей состоит в использовании для вычисления $\mathbf{Ax}^{(k)}$ и $\mathbf{b} - \mathbf{Ax}^{(k)}$ арифметики удвоенной точности.

2. Вычисляют решение системы $\mathbf{Ae}^{(k)} = \mathbf{r}^{(k)}$. Так как матрица $\mathbf{A}$ не меняется, то получение очередного приближения с использованием однаж-

ды вычисленного LU-разложения матрицы **A** (с учетом необходимых перестановок строк) требует сравнительно небольшого числа (примерно N^2) арифметических действий.

3. Вычисляют $\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)} + \mathbf{e}^{(k)}$.

Если число обусловленности не очень велико (например, $\mathrm{cond}(A) << \varepsilon_M^{-1}$), то метод довольно быстро сходится. Сходимость характеризуется постепенным установлением значащих цифр в приближениях $\mathbf{x}^{(k)}$. Если же процесс расходится, то в приближениях не устанавливаются даже старшие значащие цифры.

Итерационное уточнение не следует рассматривать как средство, позволяющее справиться с плохой обусловленностью системы. Его применение для решения плохо обусловленных систем целесообразно только, если матрица **A** и вектор правых частей **b** в компьютере заданы точно или с повышенной точностью.

Необходимо отметить одну интересную особенность. В процессе итерационного уточнения невязки $\mathbf{r}^{(k)}$ обычно не уменьшаются, а даже несколько возрастают по величине.

Кроме того, как оказывается, приближения $\mathbf{x}^{(k)}$ могут быть использованы для грубого по порядку оценивания естественного числа обусловленности $\nu_\delta(\mathbf{x})$. Для этого достаточно воспользоваться приближенной формулой

$$\nu_\delta(\mathbf{x}) \approx \varepsilon_M^{-1} \|\mathbf{e}^{(k)}\| / \|\mathbf{x}^{(k)}\|. \qquad (2.71)$$

Пример 2.19. Используя алгоритм итерационного уточнения, найдем решение системы

$$\begin{aligned} 1.03x_1 + 0.991x_2 &= 2.51, \\ 0.991x_1 + 0.943x_2 &= 2.41 \end{aligned} \qquad (2.72)$$

на 3-разрядном десятичном компьютере, считая, что режим вычислений с удвоенной точностью на нем эквивалентен использованию 6-разрядного десятичного компьютера.

Вычислим множитель первого (и в данном случае последнего) шага прямого хода метода Гаусса: $t = 0.991 / 1.03 \approx 0.962$. Так как при вычислении на 3-разрядном компьютере имеем $0.943 - 0.991 \cdot 0.962 \approx -0.01$ и $2.41 - 2.51 \cdot 0.962 \approx 0$, то второе уравнение системы (2.72) приводится к виду $-0.01x_2 = 0$.

Обратная подстановка дает приближенное решение $x_1^{(0)} = 2.24$, $x_2^{(0)} = 0$.

Итерационное уточнение.

1-й шаг. Вычислим (с удвоенной точностью) компоненты невязки:

$$r_1^{(0)} = 2.51 - 1.03x_1^{(0)} - 0.991x_2^{(0)} = -0.00320,$$
$$r_2^{(0)} = 2.41 - 0.991x_1^{(0)} - 0.943x_2^{(0)} = -0.00840.$$

Вычислив методом Гаусса решение системы

$$1.03e_1^{(0)} + 0.991e_2^{(0)} = -0.00320,$$
$$0.991e_1^{(0)} + 0.943e_2^{(0)} = -0.00840,$$

получим $e_1^{(0)} = -0.481$, $e_2^{(0)} = 0.496$. Завершается 1-й шаг вычислением $x_1 = x_1^{(0)} + e_1^{(0)}$, $x_2 = x_2^{(0)} + e_2^{(0)}$, приводящим на 3-разрядном компьютере к значениям $x_1^{(1)} = 1.96$, $x_2^{(1)} = 0.496$.

2-й шаг дает значения $r_1^{(1)} = -0.000336$, $r_2^{(1)} = -0.000088$, $e_1^{(1)} = 0.0223$, $e_2^{(1)} = -0.0235$ и $x_1^{(2)} = 1.98$, $x_2^{(2)} = 0.473$.

3-й шаг дает значения $r_1^{(2)} = 0.00186$, $r_2^{(1)} = 0.00178$, $e_1^{(2)} = 0.00084$, $e_2^{(2)} = 0.001$ и $x_1^{(3)} = 1.98$, $x_2^{(3)} = 0.474$.

Сравнивая $\mathbf{x}^{(2)}$ и $\mathbf{x}^{(3)}$, замечаем, что последние значащие цифры практически установились и, следовательно, процесс следует завершить, приняв $x_1 \approx 1.98$, $x_2 \approx 0.474$.

Использование формулы (2.71) с $\varepsilon_M = 5 \cdot 10^{-4}$ дает следующую оценку естественного числа обусловленности:

$$\nu_\delta(\mathbf{x}) \approx \varepsilon_M^{-1} \|\mathbf{e}^{(0)}\|_\infty / \|\mathbf{x}^{(0)}\|_\infty \approx 2 \cdot 10^3 \cdot 0.496 / 2.44 \approx 409.$$

Приведем для сравнения действительные значения решения и естественного числа обусловленности: $x_1 \approx 1.9812$, $x_2 \approx 0.4735$ $\nu_\delta(\mathbf{x}) \approx 273$.

2.19. Сингулярное разложение матрицы

2.19.1. *Переопределенная система*

Предположим, что требуется решить систему N алгебраических уравнений с M неизвестными, где $M < N$

$$\begin{array}{ccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \ldots & + & a_{1M}x_M & = & b_1, \\
a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & \ldots & + & a_{2M}x_M & = & b_2, \\
a_{31}x_1 & + & a_{32}x_2 & + & a_{33}x_3 & + & \ldots & + & a_{3M}x_M & = & b_3, \\
\ldots\ldots & & \ldots\ldots & & \ldots\ldots & & \ldots\ldots & & \ldots\ldots & \ldots & \ldots \\
a_{N1}x_1 & + & a_{N2}x_2 & + & a_{N3}x_3 & + & \ldots & + & a_{NM}x_M & = & b_N.
\end{array} \qquad (2.73)$$

Так как число уравнений превышает число неизвестных, то вполне вероятно, что рассматриваемая система не имеет решения. Однако, хотя

уравнения системы нельзя удовлетворить точно, можно попытаться удовлетворить их как можно точнее, минимизируя величину вектора невязки $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$. Выбор в качестве минимизируемой величины евклидовой нормы невязки приводит к методу наименьших квадратов решения переопределенных систем линейных уравнений.

2.19.2. *Сингулярное разложение матрицы*

Пусть $\mathbf{A}$ – вещественная $N \times M$ матрица размера, где $N \geq M$. Тогда существуют такие ортогональные матрицы $\mathbf{U}$ и $\mathbf{V}$ размера $N \times N$ и $M \times M$ соответственно, что

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{2.74}$$

где

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & 0 & \ldots & 0 \\ 0 & 0 & \sigma_3 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & \sigma_M \\ 0 & 0 & 0 & \ldots & 0 \\ \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

диагональная матрица размера $N \times M$ с элементами $\sigma_1 \geq \sigma_2 \geq \sigma_3 \ldots \geq \sigma_M \geq 0$ на главной диагонали.

Разложение (2.74) называется сингулярным разложением матрицы $\mathbf{A}$ или SVD-разложением[1]. Числа $\sigma_1, \ldots, \sigma_M$ называются сингулярными числами матрицы $\mathbf{A}$. Эти числа определяются по матрице $\mathbf{A}$ однозначно и являются важными ее характеристиками. Например, число ненулевых сингулярных чисел совпадает с рангом матрицы.

Сингулярное разложение существует более 100 лет. Оно было независимо открыто в 1873 г. Бельтрами[2] и в 1874 г. Жорданом для квадратных матриц. В 30-е гг. XX в. оно было распространено на прямоугольные матрицы. В вычислительную практику SVD вошло в 60-е гг. и стало важнейшим инструментом вычислений.

---

[1] От английского singular value decomposition.

[2] Эуджению Бельтрами (1835–1900) – итальянский математик. Основные труды по геометрии. Показал, что геометрия Лобачевского (планиметрия) реализуется на поверхности, называемой псевдосферой.

2.19.3. *Использование сингулярного разложения для решения переопределенных систем*

Пусть разложение (2.74) получено. Пусть ранг **A** равен r, т.е. пусть $\sigma_1 \geq \ldots \sigma_r > 0$ и $\sigma_{r+1} = \ldots = \sigma_M = 0$. Нашей целью является минимизация величины $\|\mathbf{r}\|_2 = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2$.

Так как матрица **U** ортогональна, то норма невязки **r** не изменится после ее умножения слева на $\mathbf{U}^T$. Таким образом,

$$\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 = \|\mathbf{U}^T(\mathbf{A}\mathbf{x}-\mathbf{b})\|_2^2 = \|\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{x}-\mathbf{U}^T\mathbf{b}\|_2^2 = \|\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{x}-\mathbf{U}^T\mathbf{b}\|_2^2.$$

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

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

Ясно, что минимум равен $(d_{r+1})^2 + \ldots + (d_N)^2$ и достигается он при $y_1 = d_1 / \sigma_1, \ldots, y_r = d_r / \sigma_r$. После того, как оказался найденным вектор **y**, осталось положить $\mathbf{x} = \mathbf{V}\mathbf{y}$. Если $r = M$, то решение найдено однозначно. Если же $r < M$, то компоненты $y_{r+1}, \ldots, y_M$ могут быть выбраны произвольным образом и, следовательно, задача имеет бесконечно много решений. Выбор $y_{r+1} = \ldots = y_M$ дает решение **x** с минимальной нормой (это решение называется псевдорешением системы (2.73)).

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

Заметим, что увеличение значения параметра σ_* приводит к увеличению нормы невязки; однако при этом уменьшается норма решения и оно становится менее чувствительным к входным данным.

2.19.4. *Дополнительная информация о сингулярном разложении*

Обозначим столбцы матрицы **U** через $\mathbf{u}_1, \ldots, \mathbf{u}_N$, а столбцы матрицы **V** – через $\mathbf{v}_1, \ldots, \mathbf{v}_N$. Векторы $\mathbf{u}_i$ и $\mathbf{v}_i$ называются соответственно левыми и правыми сингулярными векторами матрицы **A**.

Равенство (2.73) можно записать в виде $\mathbf{A}\mathbf{V} = \mathbf{U}\boldsymbol{\Sigma}$ или $\mathbf{A}^T\mathbf{U} = \mathbf{V}\boldsymbol{\Sigma}^T$. Сравнивая эти матричные равенства по столбцам, видим, что сингулярные векторы обладают следующими свойствами:

$$\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i,\ \mathbf{A}^T\mathbf{u}_i = \sigma_i\mathbf{v}_i,\ 1 \leq i \leq M.$$

Из этих равенств следует, в частности, что

$$\mathbf{A}^T\mathbf{A}\mathbf{v}_i = \sigma_i^2\mathbf{v}_i,\ 1 \le i \le M.$$

Таким образом, квадраты сингулярных чисел матрицы **A** совпадают с собственными значениями матрицы $\mathbf{A}^T\mathbf{A}$. Для симметричных матриц сингулярные числа просто равны модулям собственных значений.

Обратим теперь внимание на норму матрицы **A**. Делая замену $\mathbf{y} = \mathbf{V}\mathbf{x}$ и учитывая, что умножение вектора на ортогональную матрицу не меняет евклидову норму, имеем:

$$\| \mathbf{A} \|_2 = \max_{\|\mathbf{x}\|_2=1} \| \mathbf{A}\mathbf{x} \|_2 = \max_{\|\mathbf{x}\|_2=1} \| \mathbf{U}^T\mathbf{A}\mathbf{x} \|_2 = \max_{\|\mathbf{x}\|_2=1} \| \mathbf{\Sigma}\mathbf{V}^T\mathbf{x} \|_2 =$$

$$= \max_{\|\mathbf{y}\|_2=1} \| \mathbf{\Sigma}\mathbf{y} \|_2 = \max_{\|\mathbf{y}\|_2=1} \sqrt{\sigma_1^2 y_1^2 + \ldots + \sigma_M^2 y_M^2} = \sigma_1 .$$

Отметим также, что отношение σ_1 / σ_M часто используется в качестве числа обусловленности матрицы **A**. Для квадратной невырожденной матрицы **A** действительно $\mathrm{cond}_2(\mathbf{A}) = \sigma_1 / \sigma_M$.

Более подробно о сингулярном разложении и его разнообразных приложениях см. в [7, 12].

2.20. Дополнительные замечания

Следует отметить, что в данной главе оказались практически не отраженными прямые методы решения СЛАУ с разреженными матрицами (исключение составляет метод прогонки). Желающим найти доступное изложение современных прямых методов, предназначенных для решения больших линейных систем с разреженными матрицами, можно посоветовать обратиться к [13]. Укажем также на книги [14–16], специально посвященные технологии разреженных матриц.

Предположим, что после того, как было найдено решение **x** системы $\mathbf{A}\mathbf{x} = \mathbf{b}$, возникла необходимость решить систему $\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{b}$ с матрицей $\tilde{\mathbf{A}}$, отличающейся от матрицы **A** несколькими элементами. Например, могло выясниться, что заданные ранее элементы содержали грубые ошибки. Возможно также, что решаемая задача такова, что в ней элементы матрицы последовательно меняются, а правая часть остается неизменной.

Разумеется, $\tilde{\mathbf{x}}$ можно найти, решив систему снова и проигнорировав полученную на предыдущем этапе информацию. Однако в рассматриваемом случае можно найти поправку $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$ к найденному ранее решению, используя всего $O(N^2)$ операций.

Пусть $\tilde{\mathbf{A}} = \mathbf{A} - \mathbf{u}\upsilon^T$, где **u** и υ – векторы размерности N. Справедлива формула Шермана-Моррисона

$$\tilde{\mathbf{A}}^{-1} = \mathbf{A}^{-1} + \alpha(\mathbf{A}^{-1}\mathbf{u})(\upsilon^T \mathbf{A}^{-1}),$$

где $\alpha = 1 / (1 - \upsilon^T \mathbf{A}^{-1}\mathbf{u})$. Применяя ее, приходим к алгоритму:

1. Систему $\mathbf{Ay} = \mathbf{u}$ решают относительно $\mathbf{y}$.
2. Систему $\mathbf{A}^T\mathbf{z} = \upsilon$ решают относительно $\mathbf{z}$.
3. Вычисляют $\alpha = 1 / (1 - \upsilon^T\mathbf{y})$, $\beta = \mathbf{z}^T\mathbf{b}$ и $\Delta\mathbf{x} = \alpha\beta\mathbf{y}$.
4. Полагают $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$.

Суммарное число операций будет действительно составлять $O(N^2)$, если для решения систем $\mathbf{Ay} = \mathbf{u}$ и $\mathbf{A}^T\mathbf{z} = \upsilon$ применить обратную подстановку с использованием LU-разложения матрицы $\mathbf{A}$, найденного ранее на этапе решения $\mathbf{Ax} = \mathbf{b}$.

В случае, когда матрица $\tilde{\mathbf{A}}$ отличается от матрицы $\mathbf{A}$ только одним элементом $\tilde{a}_{ij} = a_{ij} + \Delta a_{ij}$, можно положить $\mathbf{u} = \Delta a_{ij}\mathbf{e}_i$, $\upsilon = \mathbf{e}_j$, а указанный алгоритм упростить следующим образом:

1. Систему $\mathbf{Ay} = \Delta a_{ij}\mathbf{e}_i$ решают относительно $\mathbf{y}$.
2. Систему $\mathbf{A}^T\mathbf{z} = \mathbf{e}_j$ решают относительно $\mathbf{z}$.
3. Вычисляют $\alpha = 1 / (1 - y_j)$, $\beta = \mathbf{z}^T\mathbf{b}$ и $\Delta\mathbf{x} = \alpha\beta\mathbf{y}$.
4. Полагают $\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$.

Основываясь на формуле Шермана–Моррисона, можно указать и способ пересчета LU-разложения матрицы. О формуле Шермана–Моррисона и ее применениях можно узнать, например, из [7, 12].

3. ИТЕРАЦИОННЫЕ МЕТОДЫ

Потребность в компьютерном моделировании всё более сложных структур привела к необходимости решения больших СЛАУ, как разреженных, так и плотных. Точные методы понятны и просты для программной реализации, однако, вычислительные затраты этих методов, которые пропорциональны N^3, серьезно ограничивают круг рассматриваемых проблем. Поэтому в последнее время широко применяются итерационные методы, и связано это со следующим. Если для сходимости итерационного метода потребуется число итераций много меньше N, то основные вычислительные затраты приходятся на умножение матрицы на вектор (один или два раза в каждой итерации в зависимости от использованного метода), т.е. в идеале уменьшаются так, что становятся пропорциональными N^2. На данный момент, наиболее эффективными и устойчивыми среди итерационных методов являются так называемые проекционные методы, и особенно тот их класс, который связан с проектированием на подпространства Крылова. Эти методы устойчивы, допускают эффективное распараллеливание и работу с предобусловливателями (о них будет сказано далее) разных типов. Достаточно полную историю развития итерационных методов можно найти в [17].

Исторически итерационные методы разрабатывались для решения разреженных СЛАУ высокого порядка. Главный источник таких СЛАУ – сеточные методы (конечных разностей, конечных объемов и конечных элементов) решения многомерных краевых задач, возникающих при моделировании процессов или явлений [18]. Позднее для ускорения итерационного процесса было предложено использовать так называемое предобусловливание, позволяющее улучшить обусловленность результирующий матрицы и, тем самым, уменьшить число итераций, требуемых для получения решения. Поскольку для хранения разреженной матрицы СЛАУ используют различные стандарты хранения, позволяющие экономить машинную память за счет не хранения нулевых элементов, то первые способы предобусловливания основывались на структуре (портрете) исходной матрицы. Позднее для стабильности и ускорения решения было предложено использовать предобусловливание для СЛАУ с плотной матрицей. Процесс перехода на многопроцессорные компьютеры обусловил поиск новых способов формирования матрицы предобусловливания, обладающих «естественным» распараллеливанием.

При решении же СЛАУ с плотной матрицей долгое время пользовались (и до сих пор пользуются) методом исключения Гаусса и его модификациями. В 1988 г. нашей соотечественницей Л.Ю. Колотилиной было

предложено использовать предфильтрацию для получения разреженной структуры, используемой для формирования предобусловливания при решении СЛАУ с плотной матрицей [19].

3.1. Классические итерационные методы и релаксация

Исторически первые итерационные методы основывались на циклическом покомпонентном изменении вектора решения, осуществляемом таким образом, чтобы обнулить соответствующий коэффициент вектора невязки и тем самым уменьшить его норму. Подобная методика уточнения решения получила название релаксация.

Хотя в настоящее время такие методы в их классической формулировке уже практически не применяются, существуют определенные классы задач, для которых разработаны их модификации, хорошо себя зарекомендовавшие. Кроме того, как будет показано далее, эти методы могут быть применены не в качестве самостоятельного средства решения СЛАУ, а для предобусловливания. Данная глава основана на [20–24].

3.1.1. *Методы Якоби и Гаусса–Зейделя*

Пусть матрица $\mathbf{A}$ системы $\mathbf{Ax} = \mathbf{b}$ такова, что ее главная диагональ не содержит нулевых элементов. Представим ее в виде разности:

$$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}, \tag{3.1}$$

где матрица $\mathbf{D}$ содержит диагональные элементы матрицы $\mathbf{A}$; матрица $\mathbf{E}$ – только поддиагональные; матрица $\mathbf{F}$ – только наддиагональные. Тогда система $\mathbf{Ax} = \mathbf{b}$ может быть записана в виде

$$\mathbf{Dx} - \mathbf{Ex} - \mathbf{Fx} = \mathbf{b}.$$

Если имеется приближение $\mathbf{x}_k$ к точному решению СЛАУ $\mathbf{x}_*$, то при $\mathbf{x}_k \neq \mathbf{x}_*$ это соотношение не выполняется. Однако, если в выражении

$$\mathbf{Dx}_k - \mathbf{Ex}_k - \mathbf{Fx}_k = \mathbf{b} \tag{3.2}$$

одно или два из вхождений вектора $\mathbf{x}_k$ заменить на $\mathbf{x}_{k+1}$ и потребовать, чтобы равенство имело место, можно получить некоторую вычислительную схему для уточнения решения.

Наиболее простой с точки зрения объема вычислений работы вариант получается при замене в (3.2) $\mathbf{Dx}_k$ на $\mathbf{Dx}_{k+1}$. При этом получается схема

$$\mathbf{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\mathbf{x}_k + \mathbf{D}^{-1}\mathbf{b}, \tag{3.3}$$

известная как метод Якоби[1] (еще называемой методом Гаусса–Якоби).

Выражение (3.3) в скалярной форме имеет вид

[1] Карл Густав Якоб Якоби (1804–1851) – немецкий математик. Известен своими трудами по вариационному исчислению, математическому анализу, дифференциальным уравнениям.

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=k+1}^{N} a_{ij} x_j^{(k)} \right) / a_{ii}, \; i = 1, \ldots, N, \qquad (3.4)$$

откуда хорошо видна основная идея метода: на (k+1)-й итерации i-й компонент вектора решения изменяется по сравнению с k-й итерацией так, чтобы i-й компонент вектора невязки $\mathbf{r}_{k+1}$ стал нулевым (при условии отсутствия изменений в других компонентах вектора **x**).

Далее приведем алгоритм метода Якоби.

Алгоритм метода Якоби

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$

Для $k = 1, 2, \ldots$

Для $i = 1, \ldots, N$

$\overline{x}_i = 0$

Для $j = 1, \ldots, i-1, i+1, \ldots, N$

$\overline{x}_i = \overline{x}_i + a_{ij} x_j^{(k-1)}$

Увеличить j

$\overline{x}_i = (b_i - \overline{x}_i) / a_{ii}$

Увеличить i

$\mathbf{x}^{(k)} = \overline{\mathbf{x}}$

Проверить сходимость и продолжить при необходимости

Увеличить k

Пример 3.1. С помощью метода Якоби решить СЛАУ

$$\begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 4 \\ 11 \end{pmatrix}$$

Пусть $\mathbf{x}^{(0)}=(1, 1)^T$, тогда получим:

1-я итерация $x_1^{(1)} = b_1 - a_{12} \cdot x_2^{(0)} / a_{11} = (4 - 1 \cdot 1) / 2 = 1.5;$ $x_2^{(1)} = b_2 - a_{21} \cdot x_1^{(0)} / a_{22} = (11 - 3 \cdot 1) / 4 = 2;$

2-я итерация $x_1^{(2)} = b_1 - a_{12} \cdot x_2^{(1)} / a_{11} = (4 - 1 \cdot 2) / 2 = 1;$ $x_2^{(2)} = b_2 - a_{21} \cdot x_1^{(1)} / a_{22} = (11 - 3 \cdot 1.5) / 4 = 1.625;$

3-я итерация $x_1^{(3)} = b_1 - a_{12} \cdot x_2^{(2)} / a_{11} = (4 - 1 \cdot 1.625) / 2 = 1.1875;$ $x_2^{(3)} = b_2 - a_{21} \cdot x_1^{(2)} / a_{22} = (11 - 3 \cdot 1) / 4 = 2;$

4-я итерация $x_1^{(4)} = b_1 - a_{12} \cdot x_2^{(3)} / a_{11} = (4 - 1 \cdot 2) / 2 = 1;$ $x_2^{(4)} = b_2 - a_{21} \cdot x_1^{(3)} / a_{22} = (11 - 3 \cdot 1.1875) / 4 \approx 1.9063;$

5-я итерация $x_1^{(5)} = b_1 - a_{12} \cdot x_2^{(4)} / a_{11} = (4 - 1 \cdot 1.9063) / 2 \approx 1.0469;$ $x_2^{(5)} = b_2 - a_{21} \cdot x_1^{(4)} / a_{22} = (11 - 3 \cdot 1) / 4 = 2;$

6-я итерация $x_1^{(6)} = b_1 - a_{12} \cdot x_2^{(5)} / a_{11} =$ $x_2^{(6)} = b_2 - a_{21} \cdot x_1^{(5)} / a_{22} =$

$= (4 - 1 \cdot 2) / 2 = 1;$ $= (11 - 3 \cdot 1.0469) / 4 \approx 1.9648;$

7-я итерация $x_1^{(7)} = b_1 - a_{12} \cdot x_2^{(6)} / a_{11} =$ $x_2^{(7)} = b_2 - a_{21} \cdot x_1^{(6)} / a_{22} =$

$= (4 - 1 \cdot 1.9648) / 2 \approx 1.0176;$ $= (11 - 3 \cdot 1) / 4 = 2;$

8-я итерация $x_1^{(8)} = b_1 - a_{12} \cdot x_2^{(7)} / a_{11} =$ $x_2^{(8)} = b_2 - a_{21} \cdot x_1^{(7)} / a_{22} =$

$= (4 - 1 \cdot 2) / 2 = 1;$ $= (11 - 3 \cdot 1.0176) / 4 \approx 1.9868;$

9-я итерация $x_1^{(9)} = b_1 - a_{12} \cdot x_2^{(8)} / a_{11} =$ $x_2^{(9)} = b_2 - a_{21} \cdot x_1^{(8)} / a_{22} =$

$= (4 - 1 \cdot 1.9868) / 2 \approx 1.0061;$ $= (11 - 3 \cdot 1) / 4 = 2;$

10-я итерация $x_1^{(10)} = b_1 - a_{12} \cdot x_2^{(9)} / a_{11} =$ $x_2^{(10)} = b_2 - a_{21} \cdot x_1^{(9)} / a_{22} =$

$= (4 - 1 \cdot 2) / 2 = 1;$ $= (11 - 3 \cdot 1.0061) / 4 \approx 1.9954.$

Очевидным недостатком схемы (3.3) – (3.4) является то, что при нахождении компонента $x_i^{(k+1)}$ никак не используется информация о пересчитанных компонентах $x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$. Исправить этот недостаток можно, переписав (3.4) в виде

$$x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=k+1}^{N} a_{ij} x_j^{(k)} \right) / a_{ii}, \; i = 1, \ldots, N, \qquad (3.5)$$

что в векторной форме эквивалентно

$$\mathbf{x}_{k+1} = (\mathbf{D} - \mathbf{E})^{-1} \mathbf{F} \mathbf{x}_k + (\mathbf{D} - \mathbf{E})^{-1} \mathbf{b}. \qquad (3.6)$$

Такая схема называется методом Гаусса–Зейделя[1] (иногда его называют просто методом Зейделя).

Приведем алгоритм метода Гаусса–Зейделя.

Алгоритм метода Гаусса–Зейделя

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$

Для $k = 1, 2, \ldots$

  Для $i = 1, \ldots, N$

    $\sigma = 0$

    Для $j = 1, \ldots, i - 1$

      $\sigma = \sigma + a_{ij} x_j^{(k)}$

    Увеличить j

    Для $j = i + 1, \ldots, N$

      $\sigma = \sigma + a_{ij} x_j^{(k-1)}$

[1] Зейдель Филипп Людвиг (1821–1896) – немецкий астроном и математик, основные работы которого относятся к математическому анализу. Одновременно и независимо от Дж. Г. Стокса Зейдель ввел понятие равномерной сходимости последовательности и ряда.

Увеличить j

$x_i^{(k)} = (b_i - \sigma) / a_{ii}$

Увеличить i

Проверить сходимость и продолжить при необходимости

Увеличить k

Пример 3.2. Решить СЛАУ (см. пример 3.1), используя метод Гаусса–Зейделя

1-я итерация $x_1^{(1)} = b_1 - a_{12} \cdot x_2^{(0)} / a_{11} = (4 - 1 \cdot 1) / 2 = 1.5;$ $x_2^{(1)} = b_2 - a_{21} \cdot x_1^{(1)} / a_{22} = (11 - 3 \cdot 3 / 2) / 4 = 1.625;$

2-я итерация $x_1^{(2)} = b_1 - a_{12} \cdot x_2^{(1)} / a_{11} = (4 - 1 \cdot 1.625) / 2 = 1.1875;$ $x_2^{(2)} = b_2 - a_{21} \cdot x_1^{(2)} / a_{22} = (11 - 3 \cdot 1.1875)/4 \approx 1.9063;$

3-я итерация $x_1^{(3)} = b_1 - a_{12} \cdot x_2^{(2)} / a_{11} = (4 - 1 \cdot 1.9063) / 2 \approx 1.0469;$ $x_2^{(3)} = b_2 - a_{21} \cdot x_1^{(3)} / a_{22} = (11 - 3 \cdot 1.0469) / 4 \approx 1.9648;$

4-я итерация $x_1^{(4)} = b_1 - a_{12} \cdot x_2^{(3)} / a_{11} = (4 - 1 \cdot 1.9648) / 2 \approx 1.0176;$ $x_2^{(4)} = b_2 - a_{21} \cdot x_1^{(4)} / a_{22} = (11 - 3 \cdot 1.0176) / 4 \approx 1.9868;$

5-я итерация $x_1^{(5)} = b_1 - a_{12} \cdot x_2^{(4)} / a_{11} = (4 - 1 \cdot 1.9868) / 2 \approx 1.0061;$ $x_2^{(5)} = b_2 - a_{21} \cdot x_1^{(5)} / a_{22} = (11 - 3 \cdot 1.0061) / 4 \approx 1.9954;$

6-я итерация $x_1^{(6)} = b_1 - a_{12} \cdot x_2^{(5)} / a_{11} = (4 - 1 \cdot 1.9954) / 2 \approx 1.0023;$ $x_2^{(6)} = b_2 - a_{21} \cdot x_1^{(6)} / a_{22} = (11 - 3 \cdot 1.0023) / 4 \approx 1.9983;$

7-я итерация $x_1^{(7)} = b_1 - a_{12} \cdot x_2^{(6)} / a_{11} = (4 - 1 \cdot 1.9983) / 2 \approx 1.0009;$ $(11 - 3 \cdot 1.0009) / 4 \approx 1.9994;$

Выражение (3.5) получается из (3.2) заменой $\mathbf{x}_k$ на $\mathbf{x}_{k+1}$ при матрицах **D** и **E**. Если вместо **D** и **E** взять **D** и **F**, то получится похожая схема

$$\mathbf{x}_{k+1} = (\mathbf{D} - \mathbf{F})^{-1}\mathbf{E}\mathbf{x}_k + (\mathbf{D} - \mathbf{F})^{-1}\mathbf{b}, \qquad (3.7)$$

которая называется обратным методом Гаусса–Зейделя.

Еще одной модификацией является симметричный метод Гаусса–Зейделя, который заключается в циклическом чередовании (3.6) и (3.7) на соседних итерациях.

Видно, что выражения (3.3), (3.4), (3.5) могут быть записаны в виде

$$\mathbf{K}\mathbf{x}_{k+1} = \mathbf{R}\mathbf{x}_k + \mathbf{b}, \qquad (3.8)$$

где матрицы **K** и **R** связаны соотношением

$$\mathbf{A} = \mathbf{K} - \mathbf{R}. \qquad (3.9)$$

Такое представление матрицы **A** называется расщеплением, а методы вида (3.8) методами, основанными на расщеплении. Очевидно, матрица

K должна быть невырожденной и легко обратимой (т.е. число затрачиваемых арифметических операций должно быть как можно меньше).

3.1.2. *Ускорение сходимости релаксационных методов*

Скорость сходимости методов, основанных на расщеплении, непосредственно связана со спектральным радиусом матрицы[1] $\mathbf{K}^{-1}\mathbf{R}$; с другой стороны, выбор **K** ограничен требованием легкой обратимости. Одним из распространенных способов улучшения сходимости является введение дополнительного параметра. Пусть ω – некоторое вещественное число. Рассмотрим вместо системы $\mathbf{Ax} = \mathbf{b}$ масштабированную систему

$$\omega\mathbf{Ax} = \omega\mathbf{b}, \tag{3.10}$$

и вместо (3.1) воспользуемся представлением

$$\omega\mathbf{A} = (\mathbf{D} - \omega\mathbf{E}) - (\omega\mathbf{F} + (1 - \omega)\mathbf{D}), \tag{3.11}$$

где матрицы **D**, **E**, **F** имеют тот же смысл, что и в (3.1).

Тогда на основании (3.10) и (3.11) можно построить итерационную схему, похожую на метод Гаусса-Зейделя,

$$(\mathbf{D} - \omega\mathbf{E})\mathbf{x}_{k+1} = (\omega\mathbf{F} + (1 - \omega)\mathbf{D})\mathbf{x}_k + \omega\mathbf{b}. \tag{3.12}$$

Схема (3.12) называется методом последовательной верхней релаксации (SOR). Для нее

$$\mathbf{K}_{\mathrm{SOR}}(\omega) = \mathbf{D} - \omega\mathbf{E}$$
$$\mathbf{R}_{\mathrm{SOR}}(\omega) = \omega\mathbf{F} + (1 - \omega)\mathbf{D}.$$

Выбор параметра ω, минимизирующего спектральный радиус, является, вообще говоря, достаточно сложной проблемой. Но для многих классов матриц такая задача исследована и оптимальные значения известны.

Приведем алгоритм метода последовательной верхней релаксации.

Алгоритм метода последовательной верхней релаксации (SOR)

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$

Для $k = 1, 2, \ldots$

  Для $i = 1, \ldots, N$

    $\sigma = 0$

    Для $j = 1, \ldots, i - 1$

      $\sigma = \sigma + a_{ij}x_j^{(k)}$

    Увеличить j

    Для $j = i + 1, \ldots, N$

      $\sigma = \sigma + a_{ij}x_j^{(k-1)}$

    Увеличить j

[1] Максимальное по абсолютной величине собственное число.

$\sigma = (b_i - \sigma)/a_{ii}$

$x_i^{(k)} = x_i^{(k-1)} + \omega(\sigma - x_i^{(k-1)})$

Увеличить i

Проверить сходимость и продолжить при необходимости

Увеличить k

Выражение (3.11) остается тождеством, если в нем поменять местами матрицы **E** и **F**. Такая перестановка дает обратный метод последовательной верхней релаксации:

$$(\mathbf{D} - \omega\mathbf{F})\mathbf{x}_{k+1} = [\omega\mathbf{E} + (1 - \omega)\mathbf{D}]\mathbf{x}_k + \omega\mathbf{b}.$$

Последовательное применение прямого и обратного методов SOR дает симметричный метод последовательной верхней релаксации (SSOR)

$$(\mathbf{D} - \omega\mathbf{E})\mathbf{x}_{k+1/2} = [\omega\mathbf{F} + (1 - \omega)\mathbf{D}]\mathbf{x}_k + \omega\mathbf{b};$$

$$(\mathbf{D} - \omega\mathbf{F})\mathbf{x}_{k+1} = [\omega\mathbf{E} + (1 - \omega)\mathbf{D}]\mathbf{x}_{k+1/2} + \omega\mathbf{b}.$$

Приведем алгоритм метода симметричной последовательной верхней релаксации.

Алгоритм метода симметричной последовательной верхней релаксации (SSOR)

Выбрать произвольное начальное приближение $\mathbf{x}^{(0)}$

$\mathbf{x}^{(1/2)} = \mathbf{x}^{(0)}$

Для $k = 1, 2, \ldots$

Для $i = 1, \ldots, N$

$\sigma = 0$

Для $j = 1, \ldots, i - 1$

$\sigma = \sigma + a_{ij}x_j^{(k-1/2)}$

Увеличить j

Для $j = i + 1, \ldots, N$

$\sigma = \sigma + a_{ij}x_j^{(k-1)}$

Увеличить j

$\sigma = (b_i - \sigma)/a_{ii}$

$x_i^{(k-1/2)} = x_i^{(k-1)} + \omega(\sigma - x_i^{(k-1)})$

Увеличить i

Для $i = N, N - 1, \ldots, 1$

$\sigma = 0$

Для $j = 1, \ldots, i - 1$

$\sigma = \sigma + a_{ij}x_j^{(k-1/2)}$

Увеличить j

Для $j = i + 1, \ldots, N$

$\sigma = \sigma + a_{ij} x_j^{(k)}$

Увеличить j

$x_i^{(k)} = x_i^{(k-1/2)} + \omega(\sigma - x_i^{(k-1/2)})$

Проверить сходимость и продолжить при необходимости

Увеличить k

3.2. Проекционные методы и подпространства Крылова

3.2.1. *Общий подход к построению проекционных методов*

Рассмотрим систему $\mathbf{Ax} = \mathbf{b}$ и сформируем для нее следующую задачу. Пусть заданы некоторые два подпространства $K \subset R^N$ и $L \subset R^N$. Требуется найти такой вектор $\mathbf{x} \in K$, который обеспечивал бы решение исходной системы, «оптимальное относительно подпространства L», т.е. чтобы выполнялось условие

$$\forall\, \mathbf{l} \in L\colon (\mathbf{Ax}, \mathbf{l}) = (\mathbf{b}, \mathbf{l}),$$

называемое условием Петрова-Галеркина. Сгруппировав обе части равенства по свойствам скалярного произведения и заметив, что $\mathbf{b} - \mathbf{Ax} = \mathbf{r}_x$, это условие можно переписать в виде

$$\forall\, \mathbf{l} \in L\colon (\mathbf{r}_x, \mathbf{l}) = 0, \tag{3.13}$$

т.е. $\mathbf{r}_x = \mathbf{b} - \mathbf{Ax} \perp L$. Такая задача называется задачей проектирования $\mathbf{x}$ на подпространство K ортогонально к подпространству L.

В более общей постановке задача выглядит следующим образом. Для исходной системы $\mathbf{Ax} = \mathbf{b}$ известно некоторое приближение $\mathbf{x}_0$ к решению $\mathbf{x}_*$. Требуется уточнить его поправку $\boldsymbol{\delta}_x \in K$ таким образом, чтобы $\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \boldsymbol{\delta}_x) \perp L$. Условие Петрова-Галеркина в этом случае можно записать в виде

$$\forall\, \mathbf{l} \in L\colon (\mathbf{r}_{\mathbf{x}_0+\boldsymbol{\delta}_x}, \mathbf{l}) = ((\mathbf{b} - \mathbf{Ax}_0) - \mathbf{A}\boldsymbol{\delta}_x, \mathbf{l}) = (\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}_x, \mathbf{l}) = 0.$$

Пусть $\dim K = \dim L = m$ (где dim – размер подпространств). Введем в подпространство K и L базисы $\{\mathbf{v}_j\}_{j=1}^m$ и $\{\boldsymbol{\omega}_j\}_{j=1}^m$ соответственно. Нетрудно видеть, что (3.13) выполняется тогда и только тогда, когда

$$\forall\, j\ (1 \le j \le m)\colon (\mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}_x, \omega_j) = 0. \tag{3.14}$$

При введении для базисов матричных обозначений $\mathbf{V} = |\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m|$ и $\mathbf{W} = |\boldsymbol{\omega}_1|\boldsymbol{\omega}_2|\ldots|\boldsymbol{\omega}_m|$ можно записать $\boldsymbol{\delta}_x = \mathbf{Vy}$ где $\mathbf{y} \subset R^m$– вектор коэффициентов. Тогда (3.14) может быть записано в виде

$$\mathbf{W}^T(\mathbf{r}_0 - \mathbf{AVy}) = 0, \tag{3.15}$$

откуда $\mathbf{W}^T\mathbf{AVy} = \mathbf{W}^T\mathbf{r}_0$ и

$$\mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0. \quad (3.16)$$

Таким образом, решение должно уточняться по формуле

$$\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{V}(\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0, \quad (3.17)$$

из которой сразу вытекает важное требование: в практических реализациях проекционных методов подпространства и их базисы должны выбираться так, чтобы $\mathbf{W}^T\mathbf{A}\mathbf{V}$ либо была малой размерности, либо имела простую структуру, удобную для обращения.

Из (3.15) также вытекает соотношение

$$\mathbf{V}\mathbf{y} = \mathbf{A}^{-1}\mathbf{r}_0 = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_0) = \mathbf{x}_* - \mathbf{x}_0,$$

т.е. $\mathbf{V}\mathbf{y} = \boldsymbol{\delta}_x$ представляет собой проекцию на подпространство K разности между точным решением и начальным приближением.

Пусть имеется набор пар подпространств $[K_i, L_i]$ ($i = 1\ldots q$) таких, что $K_1 \oplus K_2 \oplus\ldots K_q \oplus = R^N$ и $L_1 \oplus L_2 \oplus\ldots L_q \oplus = R^N$ (запись $K \oplus L$ обозначает симметричную разность множеств K и L, т.е. элемент принадлежит $K \oplus L$, если он находится либо в K, либо в L, но не в пересечении K и L, т.е. $K \oplus L = K \cup L - K \cap L$). Тогда очевидно, что последовательное применение описанного ранее процесса ко всем таким парам приведет к решению $\mathbf{x}_q$, удовлетворяющему исходной системе $\mathbf{A}\mathbf{x} = \mathbf{b}$. Соответственно, в общем виде алгоритм любого метода проекционного класса может быть записан следующим образом:

Начало
Выбор пары подпространств K и L
Построение для K **и** L **базисов** $\mathbf{V} = |\boldsymbol{\nu}_1|\boldsymbol{\nu}_2|\ldots|\boldsymbol{\nu}_m|$ и $\mathbf{W} = |\boldsymbol{\omega}_1|\boldsymbol{\omega}_2|\ldots|\boldsymbol{\omega}_m|$
$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
$\mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}$
$\mathbf{x} = \mathbf{x} + \mathbf{V}\mathbf{y}$
Продолжать до достижения сходимости

3.2.2. *Случай одномерных подпространств K и L*

Наиболее простой случай, когда пространства K и L одномерны. Пусть K и L подпространства, являющиеся множествами векторов, представимых в виде линейной комбинации векторов $\{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2,\ldots\}$ и $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2,\ldots\}$, т.е. $K = \mathrm{span}\{\boldsymbol{\nu}\}$ и $L = \mathrm{span}\{\boldsymbol{\omega}\}$. Тогда (3.17) принимает вид

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k\boldsymbol{\nu}_k, \quad (3.18)$$

причем коэффициент γ_k легко находится из условия ортогональности $\mathbf{r}_k - \mathbf{A}(\gamma_k\boldsymbol{\nu}_k) \perp \boldsymbol{\omega}_k$:

$$(\mathbf{r}_k - \mathbf{A}\gamma_k\boldsymbol{\nu}_k, \boldsymbol{\omega}_k) = (\mathbf{r}_k, \boldsymbol{\omega}_k) - \gamma_k(\mathbf{A}\boldsymbol{\nu}_k, \boldsymbol{\omega}_k) = 0,$$

откуда $\gamma_k = (\mathbf{r}_k, \boldsymbol{\omega}_k) / (\mathbf{A}\boldsymbol{\nu}_k, \boldsymbol{\omega}_k)$.

Если выбрать $\boldsymbol{\nu}_k = \boldsymbol{\omega}_k = \mathbf{r}_k$, тогда (3.18) принимает вид

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{r}_k(\mathbf{r}_k, \mathbf{r}_k) / (\mathbf{A}\mathbf{r}_k, \mathbf{r}_k). \tag{3.19}$$

Поскольку выражение в знаменателе представляет собой квадратичную форму $\mathbf{r}_k{}^T\mathbf{A}\mathbf{r}_k$, сходимость процесса гарантирована, если матрица $\mathbf{A}$ является симметричной и положительно определенной. Данный метод называется методом наискорейшего спуска. В практических задачах метод наискорейшего спуска обладает достаточно медленной сходимостью.

При выборе $\boldsymbol{\nu}_k = \mathbf{A}^T\mathbf{r}_k$ и $\boldsymbol{\omega}_k = \mathbf{A}\boldsymbol{\nu}_k$ формула (3.18) примет вид

$$\begin{aligned}\mathbf{x}_{k=1} &= \mathbf{x}_k + \mathbf{A}^T\mathbf{r}_k(\mathbf{r}_k, \mathbf{A}\mathbf{A}^T\mathbf{r}_k) / (\mathbf{A}\mathbf{A}^T\mathbf{r}_k, \mathbf{A}\mathbf{A}^T\mathbf{r}_k) = \\ &= \mathbf{x}_k + \mathbf{A}^T\mathbf{r}_k(\mathbf{A}^T\mathbf{r}_k, \mathbf{A}^T\mathbf{r}_k) / (\mathbf{A}^T\mathbf{A}\mathbf{A}^T\mathbf{r}_k, \mathbf{A}^T\mathbf{r}_k).\end{aligned} \tag{3.20}$$

Данный метод называется методом наискорейшего уменьшения невязки; условием его сходимости является невырожденность матрицы $\mathbf{A}$. Сравнивая (3.19) и (3.20), нетрудно убедится, что метод наискорейшего уменьшения невязки совпадает с методом наискорейшего спуска, примененным к системе $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

Если положить $K = L$ и на различных итерациях в качестве вектора $\boldsymbol{\nu}_k$ циклически с повторением выбирать единичные орты $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N, \mathbf{e}_1, \ldots$ то получится рассмотренный ранее метод Гаусса-Зейделя (обратный порядок выбора соответствует обратному методу Гаусса-Зейделя).

Выбор на k-й итерации $\boldsymbol{\nu}_k = \boldsymbol{\omega}_k = \mathbf{A}^T\mathbf{e}_k$ дает, так называемый, ABS-класс методов. Чтобы для различных итераций выполнилось условие $K_i \perp K_j$, возможно либо хранение всех $\boldsymbol{\nu}_k$ с их ортогонализацией по мере нахождения (схема Хуанга), либо пересчет матрицы $\mathbf{A}$ (схема Абрамова[1]). Первый вариант ведет к увеличению расходов памяти, второй – к изменению заполненности матрицы. Поэтому, данные методы пригодны лишь для решения небольших систем; с другой стороны, их единственным условием сходимости является существование решения как такового, а потому данный класс пригоден для СЛАУ с вырожденными и неквадратными матрицами. Непосредственно из определения векторов $\boldsymbol{\nu}_k$ следует, что в данных методах на k-й итерации поправка к решению вычисляется из условия обращения k-го уравнения в тождество. Если предполагать, что матрица $\mathbf{A}$ квадратная и невырожденная, вычислительная схема имеет следующий вид (метод ABR1ORT):

[1] Абрамов Александр Александрович (родился 14 февраля 1926) – профессор кафедры высшей математики Московского физико-технического института; заслуженный деятель науки Российской Федерации; Главный научный сотрудник отдела вычислительных методов Вычислительного центра им. А.А. Дородницина РАН. Профессор Абрамов является специалистом в области математики, вычислительных методов и их приложений.

Алгоритм метода ABR1ORT

Выполнять для $i = 1,\ldots, N$

$$\mathbf{x} = \mathbf{x} + \frac{b_i}{(\mathbf{a}_{i*},\mathbf{a}_{i*})} \cdot \mathbf{a}_{i*}^T$$

Выполнять для $j = i + 1,\ldots, N$

$$\alpha = \frac{(\mathbf{a}_{i*},\mathbf{a}_{j*})}{(\mathbf{a}_{i*},\mathbf{a}_{i*})}$$

$$\mathbf{a}_{j*} = \mathbf{a}_{j*} - \alpha \cdot \mathbf{a}_{i*}$$

$$b_j = b_j - \alpha \cdot b_i$$

увеличить j

увеличить i

3.2.3. *Два выбора подпространств*

В разделе 3.2.2 были рассмотрены метод наискорейшего спуска, в котором подпространства K и L были связаны соотношением $K = L$, и метод наискорейшего уменьшения невязки, который основан на соотношении $L = \mathbf{A}K$. Сами подпространства являлись одномерными, в качестве базиса K выступал вектор невязки. В этих случаях задача проектирования эквивалентна задаче минимизации функционалов. Как оказывается, подобные утверждения справедливы и в гораздо более общих случаях, которые имеют большое значение при построении более сложных и эффективных методов.

Так, если матрица $\mathbf{A}$ симметрична и положительно определена, то задача проектирования решения СЛАУ $\mathbf{Ax} = \mathbf{b}$ на любое подпространство K ортогонально к нему самому (т.е. ортогонально к пространству $L = K$) является эквивалентной задаче минимизации $\|\mathbf{x} - \mathbf{x}_*\|_{\mathbf{A}}^2$ на пространстве K. Для произвольной невырожденной матрицы $\mathbf{A}$ задача проектирования решения СЛАУ $\mathbf{Ax} = \mathbf{b}$ на любое подпространство K ортогонально подпространству $L = \mathbf{A}K$, эквивалентна задаче минимизации $\|\mathbf{r}_x\|_2^2$ на подпространстве K.

3.2.4. *Подпространства Крылова*

При построении и реализации проекционных методов важную роль играют так называемые подпространства Крылова, часто выбираемые в качестве K. Подпространством Крылова размерности m, порожденным вектором $\boldsymbol{\nu}$ и матрицей $\mathbf{A}$, называется линейное пространство

$$K_m(\boldsymbol{\nu}, \mathbf{A}) = \mathrm{span}\{\boldsymbol{\nu}, \mathbf{A}\boldsymbol{\nu}, \mathbf{A}^2\boldsymbol{\nu}, \ldots, \mathbf{A}^{m-1}\boldsymbol{\nu}\}. \qquad (3.21)$$

В качестве вектора $\boldsymbol{\nu}$ обычно выбирается невязка начального приближения $\mathbf{r}_0$; тогда выбор подпространства L и способ построения базисов подпространств полностью определяет вычислительную схему метода.

К идее использования подпространств Крылова[1] можно прийти, например, следующим образом. При построении релаксационных методов использовалось представление матрицы $\mathbf{A}$ в виде $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$. Было также показано, что методы Якоби–Зейделя являются частными случаями класса методов, основанного на расщеплении $\mathbf{A}$ в виде разности (3.9) двух матриц $\mathbf{K}$ и $\mathbf{R}$. Тогда система $\mathbf{Ax} = \mathbf{b}$ может быть записана в виде

$$\mathbf{Kx} = \mathbf{b} + \mathbf{Rx} = \mathbf{b} + (\mathbf{K} - \mathbf{A})\mathbf{x},$$

что позволяет построить итерационный процесс

$$\mathbf{K}\mathbf{x}_{k+1} = \mathbf{K}\mathbf{x}_k + (\mathbf{b} - \mathbf{A}\mathbf{x}_k),$$

или, что то же самое,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{K}^{-1}\mathbf{r}_k. \tag{3.22}$$

Если выбрать $\mathbf{K} = \mathbf{I}$ и $\mathbf{R} = \mathbf{I} - \mathbf{A}$, процесс (3.22) сводится к виду

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{r}_k, \tag{3.23}$$

откуда следует

$$\mathbf{x}_k = \mathbf{x}_0 + \mathbf{r}_0 + \mathbf{r}_1 + \ldots + \mathbf{r}_{k-1}. \tag{3.24}$$

Умножив обе части (3.23) слева на $(-\mathbf{A})$ и прибавив к ним $\mathbf{b}$, получим

$$\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}_k - \mathbf{A}\mathbf{r}_k = \mathbf{r}_k - \mathbf{A}\mathbf{r}_k,$$

что позволяет найти выражение для невязки на k-ой итерации через невязку начального приближения:

$$\mathbf{r}_k = (\mathbf{E} - \mathbf{A})\mathbf{r}_{k-1} = (\mathbf{E} - \mathbf{A})^k\mathbf{r}_0. \tag{3.25}$$

После подстановки (3.25) в (3.24) получается

$$\mathbf{x}_k = \mathbf{x}_0 + \left[\sum_{j=0}^{k-1}\left(\mathbf{E}-\mathbf{A}\right)^j\right]\mathbf{r}_0,$$

т.е. $\boldsymbol{\delta}_k \in \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0\} = K_k(\mathbf{r}_0, \mathbf{A})$. Из (3.25) следует, что в методах, использующих подпространства Крылова, невязка на k-ой итерации выражается через начальную невязку некоторым матричным полиномом.

---

[1] Алексей Николаевич Крылов (1863–1945) – русский и советский кораблестроитель, специалист в области механики, математик, академик АН СССР (1916; член-корреспондент 1914), лауреат Сталинской премии (1941), Герой Социалистического Труда (1943). В 1931 г. Крылов опубликовал работу на тему, известную теперь как подпространство Крылова или методы подпространства Крылова. Работа касалась проблем собственных значений. Крылов коснулся эффективности вычислений и подсчитал вычислительные затраты как количество «отдельных операций перемножения» – явление нетипичное для математической публикации 1931 г. Крылов представил свой собственный метод, который стал лучшим из известных к тому времени методов и широко используется до сих пор.

3.2.5. *Базис подпространства Крылова. Ортогонализация Арнольди*

Для построения базиса в пространстве Крылова $K_m(\mathbf{v}_1, \mathbf{A})$ можно применить следующий подход. Сначала находят векторы $\boldsymbol{\omega}_1 = \mathbf{v}_1$, $\boldsymbol{\omega}_2 = \mathbf{A}\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_3 = \mathbf{A}^2\boldsymbol{\omega}_1 = \mathbf{A}\boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_m = \mathbf{A}^{m-1}\boldsymbol{\omega}_1 = \mathbf{A}\boldsymbol{\omega}_{m-1}$. По определению (3.21)

$$K_m(\mathbf{v}_1, \mathbf{A}) = \operatorname{span}\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_m\}.$$

Далее переходят от $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_m\}$ к $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$, применив процедуру ортогонализации

$$\mathbf{v}_{k+1} = \boldsymbol{\omega}_{k-1} - \sum_{i=1}^{k} \alpha_i \mathbf{v}_i \tag{3.26}$$

и затем пронормировав полученные векторы. Предполагая, что предыдущие k векторов уже построены, т.е.

$$\forall\, i, j\ (1 \le i,\ j \le k)\colon (\mathbf{v}_i, \mathbf{v}_j) = \begin{cases} 0, & i \ne j \\ 1, & i = j \end{cases}, \tag{3.27}$$

(3.26) можно записать как

$$\mathbf{v}_{k+1} = \mathbf{A}\mathbf{v}_{k+1} - \sum_{i=1}^{k} \alpha_i \mathbf{v}_i .$$

Для выполнения условия ортогональности $\mathbf{v}_{k+1}$ ко всем предыдущим векторам умножают это равенство скалярно на $\mathbf{v}_j$ $(j \le k)$ и приравнивают результат к нулю

$$(\mathbf{A}\mathbf{v}_k, \mathbf{v}_j) - \sum_{i=1}^{k} \alpha_i\, (\mathbf{v}_i\, \mathbf{v}_j) = 0. \tag{3.28}$$

С учетом (3.27) получается выражение для коэффициентов α_j

$$\alpha_j = (\mathbf{A}\mathbf{v}_k, \mathbf{v}_j).$$

Описанный метод может быть оформлен в виде следующей процедуры, называемой ортогонализацией Арнольди [25].

Алгоритм ортогонализации Арнольди

Входные данные: $\mathbf{v}_1$, **такой, что** $\|\mathbf{v}_1\|_2 = 1$; $\mathbf{A}$; m

Выполнять для $j = 1, \ldots, m$

$\boldsymbol{\omega} = \mathbf{A}\mathbf{v}_j$

Выполнять для $i = 1, \ldots, j$

$h_{ij} = (\boldsymbol{\omega}, \mathbf{v}_i)$

$\boldsymbol{\omega} = \boldsymbol{\omega} - h_{ij}\mathbf{v}_i$

увеличить i

$h_{j+1,j} = \|\boldsymbol{\omega}\|_2$

Если $h_{j+1,j} = 0$, **то КОНЕЦ**. **Базис построен**.

$\mathbf{v}_{j+1} = \boldsymbol{\omega} / h_{j+1,j}$

увеличить j

Для коэффициентов ортогонализации здесь введена двойная индексация, с учетом которой внутренний цикл алгоритма алгебраически записывается как

$$h_{j+1,j}\mathbf{v}_{j+1} = \mathbf{A}\,\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i\,. \tag{3.29}$$

Коэффициенты ортогонализации $h_{i,j}$ можно объединить в виде матрицы $\mathbf{H}$, дополнив в ней недостающие позиции нулями. При этом, как видно из алгоритма, для заданной размерности пространства m генерируется m+1 векторов. Последний вектор $\mathbf{v}_{m+1}$ (возможно, нулевой) в матричном представлении означает расширение базиса $\mathbf{V}$ одним дополнительным столбцом, т.е.

$$\mathbf{V}_m = [\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m];$$
$$\mathbf{V}_{m+1} = [\mathbf{v}_1|\mathbf{v}_2|\ldots|\mathbf{v}_m|\mathbf{v}_{m+1}].$$

Соответствующий вектору $\mathbf{v}_{m+1}$ коэффициент $h_{m+1,m}$ означает расширение матрицы $\mathbf{H}$ одной дополнительной строкой (возможно, нулевой).

Пусть $\overline{\mathbf{H}}_m$ – это матрица $(m + 1) \times m$ коэффициентов $h_{m+1,m}$, а матрица $\mathbf{H}_m$ – та же самая матрица без последней строки, имеющая размерность $m \times m$. Тогда непосредственно из описания алгоритма Арнольди и (3.29) следует, что матрица $\mathbf{H}_m$ является матрицей в верхней форме Хессенберга, (матрица называется верхней хессенберговой или верхней почти треугольной, если ее ненулевые элементы расположены только в ее верхнем треугольнике, на диагонали и на поддиагонали) и для нее справедливы соотношения

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_{m+1}\overline{\mathbf{H}}_m = \mathbf{V}_m\mathbf{H}_m + \boldsymbol{\omega}_m\mathbf{e}_m{}^T; \tag{3.30}$$
$$\mathbf{V}_m{}^T\mathbf{A}\mathbf{V}_m = \mathbf{H}_m. \tag{3.31}$$

Кроме того, вследствие ортонормальности базиса $\{\mathbf{v}_j\}$ имеет место равенство

$$\mathbf{V}^T\mathbf{v}_k = \mathbf{e}_k. \tag{3.32}$$

Пример 3.3. Пусть $m = 2$, а матрица $\mathbf{A}$ и вектор $\mathbf{v}_1$ равны

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}; \quad \mathbf{v}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Применение алгоритма Арнольди дает следующее.

$\mathbf{v}_1 = (0, 1, 0)^T$
при $j = 1$ $\boldsymbol{\omega} = \mathbf{A}\mathbf{v}_1 = (1, 1, 0)^T$
при $i = 1$ $h_{11} = 1$, $\boldsymbol{\omega} = (1, 1, 0)^T - (0, 1, 0)^T = (1, 0, 0)^T$
$h_{21} = \|\boldsymbol{\omega}\|_2 = 1$
$\mathbf{v}_2 = \boldsymbol{\omega} = (1, 0, 0)^T$
при $j = 2$ $\boldsymbol{\omega} = \mathbf{A}\mathbf{v}_2 = (1, 0, 0)^T$
при $i = 1$ $h_{12} = 0$, $\boldsymbol{\omega} = (1, 0, 0)^T$
при $i = 2$ $h_{22} = 1$, $\boldsymbol{\omega} = (1, 0, 0)^T - (1, 0, 0)^T = (0, 0, 0)^T$
$h_{32} = \|\boldsymbol{\omega}\|_2 = 0$
$\mathbf{v}_3 = \boldsymbol{\omega} = (0, 0, 0)^T$

Результатом является базис из векторов $\mathbf{v}_1 = (0, 1, 0)^T$ и $\mathbf{v}_2 = (1, 0, 0)^T$, а во введенных ранее матричных обозначениях:

$$\mathbf{V}_{m+1} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \ \mathbf{V}_m = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix};$$

$$\bar{\mathbf{H}}_m = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 0 \end{pmatrix}; \ \mathbf{H}_m = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

3.2.6. *Биортогонализация Ланцоша*

Алгоритм ортогонализации Арнольди, рассмотренный ранее, для построения каждого нового вектора $\mathbf{v}_k$ требует нахождения $(k-1)$ скалярных произведений и столько же операций линейного комбинирования. Однако, как оказывается, при отказе от требования ортогональности базиса в пользу некоторого более общего условия можно построить процедуру меньшей сложности.

Системы векторов $\{\mathbf{x}_i\}_{i=1}^m$ и $\{\mathbf{y}_i\}_{i=1}^m$ называются биортогональными, если скалярное произведение $(\mathbf{x}_i, \mathbf{y}_i)$ обращается в ноль при $i \neq j$. В случае $\mathbf{x}_i \equiv \mathbf{y}_i$ условие биортогональности сводится к обычному условию ортогональности. Пусть векторы $\mathbf{v}_1$ и $\boldsymbol{\omega}_1$ таковы, что $(\mathbf{v}_1, \boldsymbol{\omega}_1) \neq 0$ и пусть системы векторов $\{\mathbf{v}_i\}_{i=1}^m$ и $\{\boldsymbol{\omega}_i\}_{i=1}^m$ определяются соотношениями:

$$\mathbf{v}_{i+1} = \mathbf{A}\mathbf{v}_i - \alpha_i \mathbf{v}_i - \beta_i \mathbf{v}_{i-1}, \ \mathbf{v}_0 \equiv 0; \tag{3.33}$$

$$\boldsymbol{\omega}_{i+1} = \mathbf{A}^T\boldsymbol{\omega}_i - \alpha_i \boldsymbol{\omega}_i - \beta_i \boldsymbol{\omega}_{i-1}, \ \boldsymbol{\omega}_0 \equiv 0; \tag{3.34}$$

$$\alpha_i = (\mathbf{A}\mathbf{v}_i, \boldsymbol{\omega}_i) / (\mathbf{v}_i, \boldsymbol{\omega}_i); \tag{3.35}$$

$$\beta_i = (\mathbf{v}_i, \boldsymbol{\omega}_i) / (\mathbf{v}_{i-1}, \boldsymbol{\omega}_{i-1}), \ \beta_1 \equiv 0. \tag{3.36}$$

Тогда

1) системы $\{\mathbf{v}_i\}_{i=1}^m$ и $\{\boldsymbol{\omega}_i\}_{i=1}^m$ являются биортогональными;

2) каждая система $\{\mathbf{v}_i\}_{i=1}^m$ и $\{\boldsymbol{\omega}_i\}_{i=1}^m$ является линейно независимой и образует базис в $K_m(\mathbf{v}_1, \mathbf{A})$ и $K_m(\boldsymbol{\omega}_1, \mathbf{A}^T)$ соответственно.

Процедура построения векторов $\mathbf{v}$ и $\boldsymbol{\omega}$ согласно формулам (3.33) – (3.36) называется биортогонализацией Ланцоша.

Очевидно, что (3.33) является частным случаем (3.29), где из всей суммы оставлено только два последних слагаемых; точно так же (3.34) является частным случаем (3.29), записанной для матрицы $\mathbf{A}^T$ и вектора $\boldsymbol{\omega}_1$. При этом коэффициенты ортогонализации в (3.33) и (3.34) одинаковы. Из сказанного следует, что можно записать аналог формулы (2.31)

$$\mathbf{W}_m{}^T\mathbf{A}\mathbf{V}_m = \mathbf{T}_m, \tag{3.37}$$

где $\mathbf{T}_m$ – симметричная трехдиагональная матрица, элементы которой определяются соотношениями

$$[\mathbf{T}_m]_{ii} = \alpha_i(\mathbf{v}_i, \boldsymbol{\omega}_i); \tag{3.38}$$

$$[\mathbf{T}_m]_{i+1,i} = [\mathbf{T}_m]_{i,i+1} = \beta_i(\mathbf{v}_i, \boldsymbol{\omega}_i), \tag{3.39}$$

выводимыми из (3.38) – (3.39).

Основным недостатком биортогонализации Ланцоша является возможность возникновения ситуации, когда $(\mathbf{v}_i, \boldsymbol{\omega}_i) = 0$; при этом продолжение процесса становится невозможным из-за неопределенности коэффициента β_{i+1}.

3.3. Предобусловливание

Вычисление итерационными методами зависит от обусловленности матрицы $\mathbf{A}$, оцениваемой числом обусловленности

$$\mathrm{cond}\mathbf{A} = \|\mathbf{A}\|\cdot\|\mathbf{A}^{-1}\| \approx |\lambda_{max}|/|\lambda_{min}|.$$

С ростом cond$\mathbf{A}$ обусловленность ухудшается, и для ряда проблем сходимость может оказаться очень медленной, и поэтому итерационный процесс может застопориться или даже оборваться, что частично будет показано ниже. Однако применение так называемого предобусловливания улучшает сходимость к требуемому решению.

Пусть $\mathbf{M}$ – некоторая невырожденная $N \times N$ матрица. Домножив $\mathbf{A}\mathbf{x} = \mathbf{b}$ на матрицу $\mathbf{M}^{-1}$, получим систему

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}, \tag{3.40}$$

которая в силу невырожденности $\mathbf{M}$ имеет то же точное решение $\mathbf{x}_*$. После обозначений $\widehat{\mathbf{A}} = \mathbf{M}^{-1}\mathbf{A}$ и $\widehat{\mathbf{b}} = \mathbf{M}^{-1}\mathbf{b}$ (3.40) примет вид

$$\widehat{\mathbf{A}}\,\mathbf{x} = \widehat{\mathbf{b}}\,. \tag{3.41}$$

Хотя (3.41) алгебраически эквивалентна $\mathbf{Ax} = \mathbf{b}$, спектральные характеристики матрицы $\widehat{\mathbf{A}}$ отличаются от характеристик матрицы $\mathbf{A}$, что ведет к изменению скорости сходимости итерационных методов для (3.41) по отношению к $\mathbf{Ax} = \mathbf{b}$ в конечной арифметике.

Процесс перехода от $\mathbf{Ax} = \mathbf{b}$ к (3.41) с целью улучшения характеристик матрицы для ускорения сходимости к решению называется предобусловливанием, а матрица $\mathbf{M}^{-1}$ – матрицей предобусловливателя. Из (3.40) сразу же вытекает важное требование: матрица $\mathbf{M}$ должна быть близка к матрице $\mathbf{A}$, чтобы $\mathrm{cond}\,\widehat{\mathbf{A}} \to 1$. Выбор $\mathbf{M} = \mathbf{A}$ сразу же приводит $\mathbf{Ax} = \mathbf{b}$ к виду $\mathbf{Ix} = \mathbf{A}^{-1}\mathbf{b}$, однако не имеет практического смысла, так как требует нахождения $\mathbf{A}^{-1}$, что, по существу, и сводится к решению $\mathbf{Ax} = \mathbf{b}$. Вторым естественным требованием является требование легкой вычислимости матрицы $\mathbf{M}$.

Невязкой системы, соответствующей вектору $\mathbf{x}$, называется вектор $\mathbf{r} = \mathbf{b} - \mathbf{Ax} = \mathbf{A}(\mathbf{x}_* - \mathbf{x})$, где $\mathbf{x}_*$ – точное решение. Невязка $\widehat{\mathbf{r}}$ системы (3.41) связана с невязкой системы $\mathbf{Ax} = \mathbf{b}$ очевидным соотношением

$$\mathbf{M}\,\widehat{\mathbf{r}} = \mathbf{r}, \tag{3.42}$$

которое справедливо и для произведений $\mathbf{z} = \mathbf{Aq}$ и $\widehat{\mathbf{r}} = \widehat{\mathbf{A}}\,\mathbf{q}$:

$$\widehat{\mathbf{z}} = \widehat{\mathbf{A}}\,\mathbf{q} = \mathbf{M}^{-1}\mathbf{Aq} \Rightarrow \mathbf{M}\,\widehat{\mathbf{z}} = \mathbf{Aq} = \mathbf{z}. \tag{3.43}$$

Это позволяет вместо явного перехода от $\mathbf{Ax} = \mathbf{b}$ к (3.41) вводить в схемы методов корректирующие шаги для учета влияния предобусловливающей матрицы (в некоторых случаях, когда матрица $\mathbf{M}$ имеет простую структуру, переход от $\mathbf{A}$ и $\mathbf{b}$ к $\widehat{\mathbf{A}}$ и $\widehat{\mathbf{b}}$ может выполняться и явно).

Из (3.43) следует еще одно условие: структура матрицы предобусловливателя должна допускать легкое и быстрое решение «обратных к предобусловливателю» систем вида $\mathbf{M}\,\widehat{\mathbf{z}} = \mathbf{z}$.

Таким образом, матрица $\mathbf{M}$ должна быть:

1) по возможности близка к матрице $\mathbf{A}$;

2) легко вычислима;

3) легко обратима.

Описанное выше предобусловливание иногда называют левым, так как домножение матрицы СЛАУ на матрицу предобусловливателя выполняется слева. Другой возможный подход основан на переходе от $\mathbf{Ax} = \mathbf{b}$ к системе

$$\mathbf{AM}^{-1}\mathbf{y} = \mathbf{b}, \tag{3.44}$$

у которой точное решение $\mathbf{y}_*$ связано с точным решением $\mathbf{x}_*$ исходной СЛАУ соотношением

$$\mathbf{x}_* = \mathbf{M}^{-1}\mathbf{y}_*. \tag{3.45}$$

Предобусловливание (3.44) реализуется путем выполнения вместо матрично-векторных умножений $\mathbf{z} = \mathbf{Aq}$ двойных умножений $\mathbf{z} = \mathbf{A}(\mathbf{M}^{-1}\mathbf{q})$. Кроме того, при достижении требуемой точности осуществляется пересчет решения в соответствии с (3.45). Подобная схема предобусловливания называется правой. Трудно сказать, какое предбусловливание выигрышно и чаще используется на практике. Далее будем считать, что используется левое предобусловливание.

3.3.1. *Виды предобусловливания*

Способы предобусловливания можно разбить на два вида: явные и неявные. Для представления системы $\mathbf{Ax} = \mathbf{b}$ с явным предобусловливанием удобно воспользоваться записью (3.40). Как было показано выше, предобусловливание может быть введено в схему метода без необходимости явного вычисления матричного произведения. Так, явное предобусловливание требует нахождения матрицы $\mathbf{M}^{-1}$ и ее умножения на вектор в каждой итерации. Для неявного необходимо решать СЛАУ с матрицей $\mathbf{M}$ в каждой итерации. Далее подробнее рассмотрим эти два вида предобусловливания.

3.3.1.1. Неявное предобусловливание

Неявно предобусловленную систему $\mathbf{Ax} = \mathbf{b}$ удобно представить в виде уравнения

$$\mathbf{MAx} = \mathbf{Mb}. \tag{3.46}$$

Большинство способов предобусловливания обоих видов основано на известном представлении матрицы в виде произведения двух матриц $\mathbf{M} = \mathbf{LU}$. О других способах неявного предобусловливания можно узнать, например, из [21, 22]. Далее рассмотрим предобусловливание, основанное на LU-разложении матрицы (см. 2.11) и классических итерационных методах (см. 3.1).

Ранее было показано, что методы Якоби и Гаусса–Зейделя, а также SOR и SSOR могут быть представлены в виде (3.8). Другой формой этого выражения является

$$\mathbf{x}_{k+1} = \mathbf{G}\mathbf{x}_k + \mathbf{f}, \tag{3.47}$$

которое получается из (3.14) домножением левой и правой частей на $\mathbf{K}^{-1}$. Проведя аналогию с (3.1) и (3.2), нетрудно увидеть, что (3.47) можно рассматривать как итерационную схему, построенную для решения СЛАУ

$$(\mathbf{I} - \mathbf{G})\mathbf{x} = \mathbf{f}, \tag{3.48}$$

в которой

$$\mathbf{f} = \mathbf{K}^{-1}\mathbf{b};$$
$$\mathbf{G} = \mathbf{K}^{-1}\mathbf{R} = \mathbf{K}^{-1}(\mathbf{K} - \mathbf{A}) = \mathbf{I} - \mathbf{K}^{-1}\mathbf{A}.$$

Таким образом, система (3.48) эквивалентна системе

$$\mathbf{K}^{-1}\mathbf{A}\mathbf{x} = \mathbf{K}^{-1}\mathbf{b},$$

т.е. предобусловленной СЛАУ $\mathbf{A}\mathbf{x} = \mathbf{b}$ с матрицей предобусловливания $\mathbf{K}$. Для методов Якоби, Гаусса–Зейделя, SOR и SSOR эти матрицы, соответственно, равны ($\mathbf{K}$ заменено на $\mathbf{M}$):

$$\mathbf{M}_{\mathrm{J}} = \mathbf{D};$$

$$\mathbf{M}_{\mathrm{GS}} = \mathbf{D} - \mathbf{E};$$

$$\mathbf{M}_{\mathrm{SOR}} = \frac{1}{\omega}(\mathbf{D} - \omega\mathbf{E});$$

$$\mathbf{M}_{\mathrm{SSOR}} = \frac{1}{\omega(2 - \omega)}(\mathbf{D} - \omega\mathbf{E})\mathbf{D}^{-1}(\mathbf{D} - \omega\mathbf{F}).$$

Скалярные множители $\frac{1}{\omega}$ и $\frac{1}{\omega(2 - \omega)}$ при практических реализациях SOR и SSOR-предобусловливания обычно не учитываются, так как дают лишь масштабирование, практически не влияющее на скорость сходимости.

При $\omega = 1$ получается симметричный метод Гаусса–Зейделя (SGS)

$$\mathbf{M}_{\mathrm{SGS}} = (\mathbf{D} - \mathbf{E})\mathbf{D}^{-1}(\mathbf{D} - \mathbf{F}).$$

Заметив, что $(\mathbf{D} - \mathbf{E})$ соответствует нижнетреугольной части матрицы $\mathbf{A}$, а $(\mathbf{D} - \mathbf{F})$ – верхнетреугольной части, получим

$$\mathbf{M}_{\mathrm{SGS}} = \mathbf{L}\mathbf{U},$$

где

$$\mathbf{L} = (\mathbf{D} - \mathbf{E})\mathbf{D}^{-1} = \mathbf{I} - \mathbf{E}\mathbf{D}^{-1},\ \mathbf{U} = \mathbf{D} - \mathbf{F}.$$

Таким образом, для решения систем вида $\mathbf{w} = \mathbf{M}_{\mathrm{SGS}}{}^{-1}\mathbf{x}$ необходимо последовательно решить

$$(\mathbf{I} - \mathbf{E}\mathbf{D}^{-1})\mathbf{z} = \mathbf{x},$$
$$(\mathbf{D} - \mathbf{F})\mathbf{w} = \mathbf{z}.$$

Известно, что если исходная матрица разреженная, то матрицы $\mathbf{L}$ и $\mathbf{U}$ являются более плотными. Основываясь на стандартном LU-разложении, матрицу предобусловливания можно формировать несколькими способами. Первый основан на нахождении $\mathbf{M}$ из $\mathbf{A}$ с помощью LU-разложения и обнуления по ходу вычислений незначительных элементов в треугольных матрицах $\mathbf{L}$ и $\mathbf{U}$ (постфильтрация). Второй, основан на предфильтрации, когда обнулением некоторых элементов исходной матрицы получают $\mathbf{A}^s$ и потом производят ее LU-разложение, тем самым формируя искомую матрицу $\mathbf{M} = \mathbf{L}\mathbf{U}$. Как было замечено выше, второй подход приводит к тому, что $\mathbf{M}$ оказывается более плотной (заполнение без ограничений), чем $\mathbf{A}^s$, что соответствует увеличению требуемой памяти.

Альтернативой полному является неполное разложение, применяемое к матрице $\mathbf{A}^s$. Данное разложение заключается в представлении

$$\mathbf{M} = \mathbf{LU} + \mathbf{R}, \qquad (3.49)$$

где **L** и **U** – как и прежде, нижне- и верхнетреугольные матрицы, соответственно, а матрица **R** является матрицей ошибки. Тогда приближенное представление $\mathbf{M} \approx \mathbf{LU}$ называется неполным LU-разложением матрицы **M** или коротко, её ILU-разложением. Самой простой версией данного разложения является ILU(0). Оно заключается в применении LU-разложения к матрице $\mathbf{A}^s$, но если $a_{ij}^s = 0$, то сразу полагается $l_{ij} = 0$ или $u_{ij} = 0$. Если $NZ(\mathbf{A}^s)$ – структура разреженности матрицы $\mathbf{A}^s$, а $NZ(\mathbf{M})$ – матрицы **M**, то очевидно, что данный способ приводит к тому, что их структуры совпадают. (Чтобы подчеркнуть тот факт, что в данной постановке задачи в структуру не вводятся новые ненулевые элементы, такую факторизацию часто называют факторизацией с нулевым заполнением.) Это удобно, когда исходная и разреженная матрицы хранятся с помощью специальных форматов хранения, учитывающих только ненулевые элементы. Далее приведен алгоритм этого разложения [22].

Алгоритм ILU(0)-разложения

Для $i = 2, \ldots, N$

  Для $k = 1, \ldots, i-1$ **и если** $(i, k) \in NZ(\mathbf{A}^s)$

    $a_{ik} = a_{ik} / a_{kk}$

    Для $j = k+1, \ldots, N$ **и если** $(i, j) \in NZ(\mathbf{A}^s)$

      $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$

    Увеличить j

  Увеличить k

Увеличить i

Поскольку полное разложение приводит к заполнению структуры матрицы без каких либо ограничений, а ILU(0)-разложение – к нулевому заполнению, очевидно, что еще одним вариантом может служить разложение с неким уровнем заполнения. Таким способом является ILU(p)-разложение. Идея его достаточно проста. Каждому ненулевому элементу матрицы $\mathbf{A}^s$ (далее в этом разделе для простоты положено, что $\mathbf{A}=\mathbf{A}^s$) первоначально присваивается нулевой уровень заполнения, в противном случае уровень принимается равным ∞. В ходе вычислений ($a_{ij}=a_{ij}-a_{ik} \cdot a_{kj}$) после вычисления элемента в позиции (i, j) необходимо провести корректировку его уровня заполнения согласно правилу

$$lev_{ij} = \min\{lev_{ij}, lev_{ik} + lev_{kj} + 1\}. \qquad (3.50)$$

Отметим, что элементы, имеющие до начала вычислений нулевой уровень, никогда его не меняют. Эти элементы определяют структуру

разреженности, совпадающую с исходной. Элементы же, получившие во время вычислений уровень $\le p$ (задаваемого значения), расширяют эту структуру. Таким образом, при $p=\infty$ данное разложение вырождается в полное, а при $p=0$ – в ILU(0). Далее приведен алгоритм ILU(p)-разложения [22].

Алгоритм ILU(p)-разложения

Для всех элементов $a_{ij} \neq 0$ задать $lev_{ij} = 0$
Для $i = 2, \ldots, N$
  Для $k = 1, \ldots, i-1$ и $lev(a_{ik}) \le p$
    $a_{ik} = a_{ik} / a_{kk}$
    Для $j = k+1, \ldots, N$
      $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$
    Увеличить j
    Провести корректировку уровня заполнения элементов *i*-й строки согласно правилу (3.50)
  Увеличить k
  Положить элементы *i*-й строки нулями, если $lev(a_{ij}) > p$
Увеличить i

Другой стратегией расширения структуры разреженности является использование постфильтрации, осуществляемой в два этапа и позволяющей контролировать уплотнение структуры [26]. Данное разложение называется неполным LU-разложением с упороживанием или ILUT. Далее приведен его алгоритм.

Алгоритм ILUT-разложения

Для $i = 1, \ldots, N$
  $\mathbf{w} = a_{i*}$
  Для $k = 1, \ldots, i-1$ и $w_k \neq 0$
    $w_k = w_k / u_{kk}$
    Применить правило обнуления к w_k
    Для $j = k+1, \ldots, N$
      $\mathbf{w} = \mathbf{w} - w_k \cdot u_{k*}$
    Увеличить j
  Увеличить k
  Применить правило обнуления к w
  $l_{ij} = w_j$, для $j = 1, \ldots, i-1$
  $u_{ij} = w_j$, для $j = i, \ldots, N$
Увеличить i

Приведем возможные правила обнуления, используемые в ILUT.

1. В строке 5 приведенного алгоритма элемент w_k обнуляется, если его значение меньше, чем евклидова норма преобразуемой строки, умноженная на задаваемый допуск обнуления τ, $0 \le \tau \le 1$.

2. В строке 10 первоначально происходит обнуление элементов строки **w**, аналогично правилу пункта 1. На втором этапе происходит сохранение по p элементов в L и U частях преобразуемой строки в сочетании с диагональными элементами, которые никогда не обнуляются.

Таким образом, пункт 2 позволяет контролировать число элементов в i-й строке. По-существу, параметр p позволяет контролировать затраты машинной памяти, а параметр τ – вычислительные затраты. Существуют различные модификации пункта 2. Одной из них является сохранение $nu(i)+p$ элементов в U части преобразуемой строки и $nl(i)+p$ элементов в L части, где $nu(i)$ и $nl(i)$ – число ненулевых элементов в U и L частях i-й строки матрицы **A**, соответственно.

Существуют и другие алгоритмы неполного LU-разложения [22, 27, 28], например, разложение ILUC, основанное не на *ikj*-версии LU-разложения, а на, так называемой, Краутовской версии и постфильтрации, описанной в ILUT.

Неявное предобусловливание хорошо зарекомендовало себя при решении СЛАУ с плотной матрицей [29–33].

3.3.1.2. Явное предобусловливание

Данный вид предобусловливания основан на нахождении матрицы $\mathbf{M}^{-1}$. Подробное рассмотрение способов формирования явного предобусловливания можно найти, например, в [22, 34–36]. Из данного вида по своей работоспособности выделяются способы, основанные на приближенном обращении (approximate inverse). Их удобно разделить на три группы.

Способы первой группы появились исторически первыми и основаны на нахождении матрицы **M**, минимизирующей норму Фробениуса матрицы невязки $\mathbf{E} - \mathbf{AM}$ [37]. Существует две стратегии вычисления обратной матрицы: первая основана на вычислении обратной матрицы «глобально»; вторая основана на постолбцовом вычислении обратной матрицы. Среди способов группы выделяются: SPAI – разреженный приближенный обратный (sparse approximate inverse), использующий постолбцовую стратегию в сочетании с QR-разложением и итерационным уточнением [38]; SPMR – минимальных невязок с самопредобусловливанием (Self-preconditioned Minimal Residual) [37], использующий постолбцовую стратегию, реализуемую с помощью предобусловленного итерационного метода минимальных невязок.

Вторую группу образуют способы, формирующие матрицу предобусловливания с помощью факторизованных приближенных обратных матриц. Эти способы основаны на неполном обращении треугольных матриц **L** и **U**, т.е. неполном нахождении обратной матрицы. Так, если существует разложение $\mathbf{A} = \mathbf{LDU}$, где **L** – нижнетреугольная матрица с единичной диагональю, **D** – диагональная матрица, а **U** – верхнетреугольная матрица с единичной диагональю, то матрица $\mathbf{A}^{-1}$ может быть найдена с помощью разложения $\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{D}^{-1}\mathbf{L}^{-1} = \mathbf{Z}\mathbf{D}^{-1}\mathbf{W}^{T}$, где $\mathbf{Z} = \mathbf{U}^{-1}$ и $\mathbf{W} = \mathbf{L}^{-T}$ – верхнетреугольные матрицы с единичной диагональю. Факторизованные приближенные обратные предобусловливатели могут быть получены вычислением с помощью разреживания матриц $\overline{\mathbf{Z}} \approx \mathbf{Z}$, $\overline{\mathbf{W}} \approx \mathbf{W}$ и $\overline{\mathbf{D}} \approx \mathbf{D}$, тогда $\mathbf{M} = \overline{\mathbf{Z}}\ \overline{\mathbf{D}}^{-1}\ \overline{\mathbf{W}}^{T} \approx \mathbf{A}^{-1}$. Имеется несколько подходов для вычисления приближенного обращения. Они не требуют нахождения треугольных матриц **L** и **U**, т.е. предобусловливание вычисляется непосредственно из матрицы **A**. Среди этого класса выделяются: метод FSAI [39] и AINV, основанный на бисопряженном процессе [36].

В третью группу входят методы, вычисляющие обратную матрицу, основываясь на двушаговом процессе. Первоначально производится неполное LU-разложение $\mathbf{A}^{s} \approx \overline{\mathbf{L}}\overline{\mathbf{U}}$ (используя стандартные способы, описанные выше), а затем полученное неполное разложение приближенно обращается. Так, обращение матриц $\overline{\mathbf{L}}$ и $\overline{\mathbf{U}}$ может быть получено, например, решением $2N$ треугольных систем вида

$$\overline{\mathbf{L}}\ \mathbf{x}_i = \mathbf{e}_i,\ \overline{\mathbf{U}}\ \mathbf{y}_i = \mathbf{e}_i\ (1 \le i \le N),$$

где $\mathbf{x}_i$, $\mathbf{y}_i$ и $\mathbf{e}_i$ – i-е столбцы обратных $\overline{\mathbf{L}}$, $\overline{\mathbf{U}}$ и единичной матрицы, соответственно. Есть и другие варианты нахождения обратных матриц [36].

3.3.2. *Выбор структуры разреженности*

Для использования предобусловливания при решении СЛАУ с плотной матрицей итерационными методами необходимо задать структуру (портрет) разреженности матрицы предобусловливания. Поскольку предобусловливание является способом уменьшения количества итераций за счет улучшения сходимости при итерационном решении СЛАУ, то выбор структуры разреженности матрицы предобусловливания является важной практической задачей. Одной из простейших стратегий является выбор заранее известной структуры, например, ленточной или блочно-диагональной, т.е. элементы матрицы $\mathbf{A}^{s}$ совпадают с ленточной (блочно-диагональной) частью матрицы **A** и равны нулю вне данной структуры. Такая предфильтрация характеризуется шириной ленты (размером блока)

и, таким образом, основана на позиции элемента в матрице (структурная предфильтрация), а не на его значении. Результаты её использования при решении плотных СЛАУ можно найти, например, в [29, 40].

В работе [41] приведено несколько способов, специально приспособленных для построения матрицы $\mathbf{M}^{-1}$. Одним из них является выбор k максимальных элементов в каждом столбце, так что в результате в матрице будет kN элементов не равных нулю. Данный способ связан с многократным поиском, поэтому он характеризуется значительными временными затратами на предфильтрацию.

Более часто используемой стратегией является динамическое (основанное на значении элемента, а не на его позиции) определение структуры разреженности с помощью некоторого порога (алгебраическая предфильтрация). Она заключается в задании (нахождении) определенного порога, с помощью которого происходит обнуление малозначащих элементов путем сравнения модуля каждого элемента с данным (полученным) значением.

Упомянутый порог обнуления, в свою очередь, может быть получен многими способами. Самым простым из них является проверка на малость модулей элементов матрицы с заданным порогом обнуления (ε):

$$a_{ij}^{s} = 0, \text{ если } \left|a_{ij}\right| < \varepsilon . \qquad (3.51)$$

Он является простейшим, с точки зрения вычислений, и требует наименьшего времени на формирование разреженной матрицы $\mathbf{A}^{s}$, поскольку порог обнуления задается непосредственно, без необходимости его вычисления.

Другой способ предфильтрации предложен Л.Ю. Колотилиной [19]:

$$a_{ij}^{s} = 0, \text{ если } \left|a_{ij} / a^{\max}\right| < \varepsilon , \qquad (3.52)$$

где a^{max} – максимальный элемент матрицы. Способ был предложен для применения явного предобусловливания при решении СЛАУ с плотной матрицей и основан на нормировке всей матрицы с помощью максимального элемента. Понятно, что при этом достаточно много времени требуется на поиск наибольшего элемента в матрице. Однако данный способ широко применяется при построении явного предобусловливания для решения СЛАУ с плотной матрицей [34, 42].

В [43] использовалось также обнуление

$$a_{ij}^{s} = 0, \text{ если } \left|a_{ij}\right| < \varepsilon = \left\|\mathrm{A}\right\|_{\infty} \tau / N , \qquad (3.53)$$

где $\left\|\mathrm{A}\right\|_{\infty} = \max_i \sum_{j=1}^{N} \left|a_{ij}\right|$, а τ – допуск обнуления. После того, как исходную систему с плотной матрицей преобразуют с помощью вейвлетов,

матрица СЛАУ остается достаточно плотной, но большая часть ее элементов имеет небольшие абсолютные значения [33, 44–46]. Далее, применяя предфильтрацию, из полученной матрицы получают требуемую матрицу $\mathbf{A}^s$. Данный способ основан на бесконечной норме матрицы или, другими словами, на наибольшей сумме элементов строк матрицы. Очевидно, что значительные временные затраты связаны с поиском наибольшей суммы строки. Отметим, что предыдущий и данный способы являются, пожалуй, самыми распространенными способами приведения матрицы к разреженному виду.

Известен способ выбора структуры разреженности, основанный на нахождении наибольшего элемента в строке [46],

$$a_{ij}^s = 0, \text{ если } \left| a_{ij} \Big/ a_i^{\max} \right| < \varepsilon , \qquad (3.54)$$

где a_i^{max} – максимальный элемент в i-й строке. Поскольку способ основан на нахождении максимального элемента в каждой строке, порог обнуления является разным для каждой из строк. После того, как максимальный элемент найден, происходит процесс, подобный описанному при использовании подхода (3.52), но вместо всей матрицы сканируется строка матрицы. Очевидно, что при использовании как данного, так и подхода (3.52), значительные затраты времени связаны с поиском максимального элемента. Как видно из приведенных (в хронологическом порядке) способов, с течением времени исследователи пытались найти предфильтрацию, адаптивную к различным задачам, возникающим перед ними.

Существуют и другие способы предфильтрации. Одни позволяют представить исходную матрицу суммой нескольких разреженных матриц, за счет использования специальных подходов, и в дальнейшем использовать структуру одной из них в качестве структуры матрицы предобусловливания (near and far) [29, 40, 47]. Другие строятся, используя знание о геометрических или топологических особенностях исследуемой задачи [34, 42].

Отметим, что алгебраическая предфильтрация может сочетаться практически со всеми другими видами, как со структурной, так и с другими способами динамической предфильтрации. Также понятно, что приведенные способы алгебраической предфильтрации могут быть использованы как постфильтрация при формировании матрицы предобусловливания. Таким образом, данный вид предфильтрации является, пожалуй, самым универсальным.

Упомянутые способы алгебраической предфильтрации можно разделить на две группы. Способы глобальной предфильтрации (3.51)–(3.53) задают общий порог обнуления для всех элементов матрицы. К способам

локальной предфильтрации относится способ (3.54), определяющий для каждой строки свой порог обнуления. Априорно трудно сказать, какой подход предпочтительнее, как с точки зрения минимизации времени решения СЛАУ, так и стабильности допуска/порога обнуления при изменении матрицы СЛАУ.

Предложенный в [48] способ локальной предфильтрации основан на идее постфильтрации, предложенной в [26], при формировании матрицы предобусловливания с помощью ILUT-разложения. Способ состоит в том, что матрица $\mathbf{A}^s$ формируется из матрицы $\mathbf{A}$ путем обнуления некоторых элементов согласно следующему правилу. Перед предфильтрацией i-й строки вычисляется ее евклидова норма, умножением которой на задаваемый допуск обнуления τ получают i-й порог обнуления ε_i. После того, как все элементы строки будут проверены на малость, необходимо повторить те же действия со следующей (i+1)-й строкой. Процесс закончится после того, как все строки матрицы $\mathbf{A}$ будут обработаны подобным образом. Данный процесс можно представить в виде формулы

$$a_{ij}^s = 0, \text{ если } \left|a_{ij}\right| < \varepsilon_i = \left\|a_{i*}\right\|_2 \tau , \tag{3.55}$$

где $\left\|a_{i*}\right\|_2 = \sqrt{\sum_{j=1}^{N} \left|a_{ij}\right|^2}$ – норма i-й строки. Таким образом, порог обнуления является своим для каждой из строк, т.е. данный способ относится к группе способов локальной предфильтрации.

Другой способ глобальной предфильтрации, предложенный в [49], заключается в следующем. Структура разреженности матрицы $\mathbf{A}^s$ определяется с помощью порога обнуления, получаемого из произведения евклидовой нормы матрицы $\mathbf{A}$ и задаваемого допуска обнуления τ. После того, как значение порога обнуления определено, происходит фильтрация исходной матрицы $\mathbf{A}$. Данный процесс описывается формулой

$$a_{ij}^s = 0, \text{ если } \left|a_{ij}\right| < \varepsilon = \left\|\mathbf{A}\right\|_{\mathrm{F}} \tau , \tag{3.56}$$

где $\left\|\mathbf{A}\right\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \left|a_{ij}\right|^2}$.

Если максимальный элемент матрицы находится на главной диагонали, то использование этого ускоряет процесс предфильтрации уже известных способов. Так, время предфильтрации (3.52) при использовании этого подхода сократится за счет поиска только на главной диагонали, а не по всей матрице. В способе (3.53) можно положить, что максимальной суммой обладает главная диагональ, т.е.

$$a_{ij}^s = 0, \text{ если } \left|a_{ij}\right| < \varepsilon = \sum_{i=1}^{N} \left|a_{ii}\right| \tau / N , \tag{3.57}$$

и не тратить время на поиски наибольшей суммы строк. Для способа (3.54) можно ограничиться условием того, что максимальные элементы в строке находятся на главной диагонали, т. е. воспользоваться правилом

$$a_{ij}^{s} = 0, \text{ если } \left| a_{ij} / a_{ii} \right| < \varepsilon . \qquad (3.58)$$

3.4. Методы крыловского типа

Ранее был построен класс проекционных методов, основой которых служит условие Петрова-Галеркина (3.13) и формула (3.16). Далее были приведены примеры построения нескольких методов для достаточно простого случая одномерных подпространств K и L. Однако выбор $m = 1$, как правило, дает медленную сходимость, и соответствующие методы оказываются пригодными лишь для СЛАУ небольшой размерности. В настоящее время широкое распространение получили проекционные методы, которые в качестве K используют описанные ранее подпространства Крылова размерности больше единицы. Такие методы называются методами крыловского типа.

3.4.1. *Метод полной ортогонализации*

Наиболее характерным примером применения описанного в 3.2.1 подхода является метод полной ортогонализации, известный также как метод Арнольди или FOM[1].

Положим $K = L$ и будем проектировать искомое решение на подпространство Крылова $K_m(\mathbf{v}_1, \mathbf{A})$, где в качестве $\mathbf{v}_1$ выступает пронормированный вектор начальной невязки

$$\mathbf{v}_1 = \mathbf{r}_0 / \beta, \ \beta = \|\mathbf{r}_0\|_2. \qquad (3.59)$$

Тогда $\mathbf{V} = \mathbf{W}$, для построения базиса можно воспользоваться ортогонализацией Арнольди (3.2.5), а матрица $\mathbf{W}^T\mathbf{A}\mathbf{V}$ в (3.31) будет иметь верхнюю хессенберговскую форму и, следовательно, окажется легко обратимой. Решение СЛАУ с такой матрицей сведется к $(m-1)$ гауссовскому исключению строк и последовательному обратному ходу.

Присутствующий в (3.17) сомножитель $\mathbf{W}^T\mathbf{r}_0$ в силу (3.59) и (3.32) будет равен $\beta\mathbf{e}_1$. Таким образом, формула (3.17) будет сведена к

$$\mathbf{x}_m = \mathbf{x}_0 + \beta \mathrm{V}\, \mathbf{H}_m^{-1}\mathbf{e}_1 . \qquad (3.60)$$

На основании (3.60) и описанного в 3.2.5 алгоритма ортогонализации Арнольди, необходимого для построения базиса $\mathbf{V}$, метод FOM может быть записан следующим образом.

---

[1] Full Orthogonalization Method.

Алгоритм метода FOM

Выбрать произвольное начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

$\beta = \|\mathbf{r}_0\|_2$; $\boldsymbol{\nu}_1 = \mathbf{r}_0 / \beta$

Создать $(m \times m)$-матрицу $\mathbf{H}_m$; **положить** $\mathbf{H}_m = 0$

Создать m-вектор $\boldsymbol{f}$; **положить** $\boldsymbol{f} = \beta\,\mathbf{e}_1$

Для $j = 1, \ldots, m$

  $\mathbf{w}_j = \mathbf{A}\,\boldsymbol{\nu}_j$

  Для $i = 1, \ldots, j$

    $h_{ij} = (\mathbf{w}_j, \boldsymbol{\nu}_i)$

    $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\boldsymbol{\nu}_i$

  увеличить i

  $h_{j+1,\,j} = \|\mathbf{w}_j\|_2$

  Если $h_{j+1,\,j} = 0$

    то $m = j$

      Прервать цикл по j

  $\boldsymbol{\nu}_{j+1} = \mathbf{w}_j / h_{j+1,\,j}$

увеличить j

Для $i = 2, \ldots, m$

  Обнулить $h_{i,\,i-1}$**, выполнив гауссовское исключение для** i-й **строки системы** $\mathbf{H}_m\mathbf{y} = \boldsymbol{f}$ **относительно** $(i-1)$-й

увеличить i

Найти $\mathbf{y}$ **из верхнетреугольной системы** $\mathbf{H}_m\mathbf{y} = \boldsymbol{f}$

$$\mathbf{x}_m = \mathbf{x}_0 + \sum_{i=1}^{m} y_i \boldsymbol{\nu}_i$$

В такой формулировке метод обладает серьезным недостатком: заранее неизвестно, какой должна быть размерность подпространства m, чтобы найденное решение $\mathbf{x}_m$ имело достаточную точность.

Проверка условия $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \geq Tol$ (где Tol – требуемая точность) может быть достаточно просто введена в алгоритм метода FOM, если гауссовское исключение строк $\mathbf{H}$ производится сразу же в цикле ортогонализации Арнольди (при этом коэффициенты исключения необходимо хранить в отдельном векторе, так как матрица $\mathbf{H}$ по мере нахождения базиса расширяется).

Однако на практике чаще применяется другой подход, гораздо более простой с точки зрения программной реализации. Его суть заключается в том, что заранее выбирается некоторая размерность подпространства m, относительно небольшая по сравнению с порядком СЛАУ. После того,

как решение $\mathbf{x}_m$ найдено, проверяется, является ли оно достаточно точным. Если необходимая точность еще не достигнута, то весь процесс повторяется с вектором $\mathbf{x}_m$ в качестве начального приближения:

Выбрать $m < N$ **и произвольное начальное приближение** $\mathbf{x}_0$

$\beta = \|\mathbf{b} - \mathbf{A}\,\mathbf{x}_0\|_2$

Начало

Алгоритм метода FOM

$\mathbf{x}_0 = \mathbf{x}_m$

Продолжать пока $\|\mathbf{b} - \mathbf{A}\,\mathbf{x}_0\|_2 / \beta \geq Tol$

Такая схема носит название перезапускаемого FOM или FOM(m). При программной реализации затраты памяти приходятся на хранение базисных векторов $\mathbf{v}_1, \ldots, \mathbf{v}_m$ подпространства Крылова; они составляют $O(mN)$ чисел.

Еще одним вариантом метода является неполная ортогонализация, когда вектор $\mathbf{w}_{j+1}$ ортогонализуется не ко всем ранее найденным $\mathbf{v}_1, \ldots, \mathbf{v}_j$, а только к k предыдущим векторам $\mathbf{v}_{j-m+1}, \ldots, \mathbf{v}_j$. Это позволяет уменьшить расходы памяти при программной реализации метода. Соответствующая схема носит название IOM[1] [22].

Далее на примере построенного метода FOM покажем, как в схему проекционного метода может быть введено предобусловливание $\mathbf{Ax} = \mathbf{b}$ без необходимости явного вычисления матричного произведения.

Перепишем алгоритм FOM в применении к системе $\mathbf{Ax} = \mathbf{b}$ с матрицей предобусловливателя $\mathbf{M}$. Очевидно, что в нем необходимо изменить те фрагменты, где фигурирует матрица СЛАУ и вектор правой части:

$\mathbf{r}_0 = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\,\mathbf{x}_0)$

……..

Для $j = 1, \ldots, N$

$\mathbf{w}_j = \mathbf{M}^{-1}\mathbf{A}\mathbf{v}_j$

Увеличить j

……..

$$\mathbf{x}_m = \mathbf{x}_0 + \sum_{i=1}^{m} y_i \mathbf{v}_i$$

Таким образом, вектор начальной невязки $\mathbf{r}_0$ и векторы $\mathbf{w}_j$ находятся из решения систем $\mathbf{Mr}_0 = \mathbf{b} - \mathbf{Ax}_0$ и $\mathbf{Mw}_j = \mathbf{Av}_j$, соответственно. Напомним, что для матрицы $\mathbf{M}$ было сформировано требование легкой обратимости. Для ILU-предобусловливания решение таких систем сводится к решению двух треугольных систем, для SSOR-предобусловливания – к

[1] Imcomplete Orthogonalization Method. (Встречается также название Truncated FOM.)

решению двух треугольных систем и умножению на диагональную матрицу и т.д.

Следовательно, алгоритм FOM с введенными в него предобусловливанием матрицей **M** можно записать следующим образом.

Алгоритм предобусловленного FOM

Построить матрицу предобусловливателя M
Выбрать произвольное начальное приближение $\mathbf{x}_0$ и
$\mathbf{z} = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
Найти $\mathbf{r}_0$ **из системы** $\mathbf{M}\mathbf{r}_0 = \mathbf{z}$
$\beta = \|\mathbf{r}_0\|_2$; $\boldsymbol{\nu}_1 = \mathbf{r}_0 / \beta$
Создать $(m \times m)$-матрицу $\mathbf{H}_m$; **положить** $\mathbf{H}_m = 0$
Создать m-вектор $\boldsymbol{f}$; **положить** $\boldsymbol{f} = \beta\,\mathbf{e}_1$
Для $j = 1, \ldots, m$
    $\mathbf{z} = \mathbf{A}\,\boldsymbol{\nu}_j$
    Найти $\mathbf{w}_j$ **из системы** $\mathbf{M}\mathbf{w}_j = \mathbf{z}$
    Для $i = 1, \ldots, j$
        $h_{ij} = (\mathbf{w}_j, \boldsymbol{\nu}_i)$
        $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\boldsymbol{\nu}_i$
    увеличить i
    $h_{j+1,\,j} = \|\mathbf{w}_j\|_2$
    Если $h_{j+1,\,j} = 0$
        то $m = j$
            Прервать цикл по j
    $\boldsymbol{\nu}_{j+1} = \mathbf{w}_j / h_{j+1,\,j}$
увеличить j
Для $i = 2, \ldots, m$
    Обнулить $h_{i,i-1}$**, выполнив гауссовское исключение для**
    i-й **строки системы** $\mathbf{H}_m\mathbf{y} = \boldsymbol{f}$ **относительно** $(i-1)$-й
увеличить i
Найти $\mathbf{y}$ **из верхнетреугольной системы** $\mathbf{H}_m\mathbf{y} = \boldsymbol{f}$

$$\mathbf{x}_m = \mathbf{x}_0 + \sum_{i=1}^{m} y_i \boldsymbol{\nu}_i$$

Подобным же образом предобусловливание может быть введено и в схему любого другого проекционного метода, требующего лишь возможности вычисления матрично-векторных произведений без пересчета элементов матицы СЛАУ.

3.4.2. *Метод обобщенных минимальных невязок*

Пусть пространства K и L связаны соотношением $L = \mathbf{A}K$, причем в качестве K используется подпространство Крылова $K_m(\boldsymbol{\nu}_1, \mathbf{A})$ с выбором $\boldsymbol{\nu}_1 = \mathbf{r}_0 / \beta$, $\beta = \|\mathbf{r}_0\|_2$. Для построения ортонормированного базиса воспользуемся ортогонализацией Арнольди. Вместо проектирования (3.16) рассмотрим эквивалентную задачу минимизации функционала $\boldsymbol{\phi}(\mathrm{x}) = \|\mathbf{r}_\mathrm{x}\|_2^2$ на пространстве K. Так как любой вектор $\mathbf{x}$ из пространства $(\mathbf{x}_0 + K_m)$ может быть записан в виде $\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}$, где $\mathbf{y}$ – вектор размера m, то задачу минимизации функционала можно переписать как

$$\mathbf{y}_m = \mathrm{argmin}_\mathrm{y} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_\mathrm{m}\mathbf{y})\|_2^2.$$

Такая вычислительная схема называется методом обобщенных минимальных невязок (GMRES) [50].

С использованием определения вектора $\boldsymbol{\nu}_1$ и соотношений (3.30), (3.32) имеем $\mathbf{r} = \mathbf{b} - \mathbf{A}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}) = \mathbf{V}_{m+1}(\boldsymbol{\beta}\, \mathbf{e}_1 - \bar{\mathbf{H}}_m\, \mathbf{y})$.

Поскольку матрица $\mathbf{V}_{m+1}$ составлена из ортонормированных столбцов, справедливо равенство

$$\boldsymbol{\psi}(\mathbf{y}) = \boldsymbol{\phi}(\mathbf{x}_0 + \mathbf{V}_m\mathbf{y}) = \|\beta\, \mathbf{e}_1 - \bar{\mathbf{H}}_m\, \mathbf{y}\|_2^2. \qquad (3.61)$$

Таким образом, для нахождения коэффициентов линейного комбинирования векторов $\{\boldsymbol{\nu}_i\}$ в GMRES необходимо решить СЛАУ

$$\bar{\mathbf{H}}_m\mathbf{y} = \beta\mathbf{e}_1, . \qquad (3.62)$$

которая является переопределенной (так как матрица $\bar{\mathbf{H}}_m$ имеет размерность $(m + 1) \times m$).

Матрица системы (3.62) имеет верхнюю хессенбергову форму. Поэтому (3.62) проще всего решать с помощью предварительного приведения к верхнетреугольному виду; последняя строка $\bar{\mathbf{H}}_m$ при этом обнуляется. С учетом хессенберговой формы наиболее простым вариантом оказываются вращения Гивенса. Всего необходимо m таких вращений $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_m$, причем $\mathbf{G}_k$ строится так, чтобы его применение к $(h_{kk}, h_{k+1,k})^T$ обнуляло второй компонент этого вектора.

Структура матриц $\mathbf{G}_1$ и $\mathbf{G}_2$ имеет вид

$$\mathbf{G}_1 = \begin{pmatrix} s & c & & & 0 \\ -c & s & & & \\ & & 1 & & \\ & & & \ddots & \\ 0 & & & & 1 \end{pmatrix} \qquad \mathbf{G}_2 = \begin{pmatrix} 1 & & & & 0 \\ & s & c & & \\ & -c & s & & \\ & & & \ddots & \\ 0 & & & & 1 \end{pmatrix}$$

Числа s и c связаны соотношением $s^2 + c^2 = 1$ (это позволяет их интерпретировать как синус и косинус некоторого угла) и определяются как:

$$t_k = \frac{h_{kk}}{h_{k+1,k}};$$

$$c_k = \frac{1}{\sqrt{1+t_k^2}};$$

$$s_k = t_k c_k.$$

Как и в случае FOM, основной проблемой в GMRES является выбор размерности m пространства K; хранение базисных векторов этого пространства определяет основные затраты памяти при программной реализации метода, составляющие $O(mN)$ чисел.

Наиболее распространенной модификацией GMRES является перезапускаемый GMRES или GMRES(m). Выбирается некоторая размерность $m < N$, определяемая, по существу, лишь доступным объемом памяти, и после обновления решения $\mathbf{x} = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$ оно принимается за новое начальное приближение, после чего процесс повторяется.

Такое обновление после построения m базисных векторов и нахождения коэффициентов их комбинирования называется GMRES-циклом, тогда как построение одного очередного вектора считается GMRES-итерацией.

Внутри цикла реализуется проверка условия завершения, которая легко реализуется, если совмещать построение базиса с помощью процедуры Арнольди и применение вращений Гивенса к уже найденным элементам матрицы $\bar{\mathbf{H}}_m$.

Далее приведен алгоритм предобусловленного метода GMRES(m) с вращениями Гивенса.

Алгоритм метода GMRES(m) с вращениями Гивенса
Построить матрицу предобусловливателя M
Выбрать $m < N$ **и произвольное начальное приближение** $\mathbf{x}_0$
$\mathbf{z} = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
$\gamma = \|\mathbf{z}\|_2$
Найти $\mathbf{r}^{(0)}$ **из системы** $\mathbf{M}\,\mathbf{r}_0 = \mathbf{z}$
$\beta = \|\mathbf{r}_0\|_2$
Создать $((m + 1) \times m)$**-матрицу H**
Создать $(m + 1)$**-вектор q**
Для $N_{it} = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}
$\mathbf{H} = 0$; $\mathbf{q} = \beta\,\mathbf{e}_1$

$\mathbf{v}_1 = \mathbf{r}_0 / \beta$
Для $i = 1, \ldots, m$
$\mathbf{z} = \mathbf{A}\mathbf{v}_i$
Найти w из системы $\mathbf{M}\mathbf{w} = \mathbf{z}$
Для $k = 1, \ldots, i$
$h_{ki} = (\mathbf{w}, \mathbf{v}_k)$
$\mathbf{w} = \mathbf{w} - h_{ki}\mathbf{v}_k$
увеличить k
$h_{i+1,i} = \|\mathbf{w}\|_2$
$\mathbf{v}_{i+1} = \mathbf{w} / h_{i+1,i}$
Для $k = 1, \ldots, i-1$
$(h_{1i}, \ldots, h_{i+1,i})^T = \mathbf{G}_k(h_{1i}, \ldots, h_{i+1,i})^T$
увеличить k
Построить $\mathbf{G}_i$ **так, что** $[\mathbf{G}_i h_{\cdot,i}]_{i+1} = 0$
$\mathbf{q} = \mathbf{G}_i\mathbf{q}$
Если $|q_{i+1}| / \gamma < Tol$
то $p = i$, **выйти из цикла по** i
увеличить i
Решить треугольную СЛАУ $\mathbf{H}_{1:p,1:p}\mathbf{y} = \mathbf{q}_{1:p}$
$\mathbf{x}_0 = \mathbf{x}_0 + \sum_{i=1}^{p} y_i \mathbf{v}_i$
$\mathbf{z} = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
Найти $\mathbf{r}_0$ **из системы** $\mathbf{M}\mathbf{r}_0 = \mathbf{z}$
$\beta = \|\mathbf{r}_0\|_2$
Продолжать пока $\beta / \gamma \geq Tol$

Очевидно, что при программной реализации нет необходимости в непосредственном хранении матриц вращения $\mathbf{G}_i$, так как каждая такая матрица полностью определяется двумя числами – косинусом и синусом угла поворота – и индексами компонент вектора, к которым применяется вращение.

3.4.3. *Метод бисопряженных градиентов*

Ранее была описана биортогонализация Ланцоша, которая в отличие от ортогонализации Арнольди, использует для построения базиса экономичные трехчленные формулы. Выберем в качестве пространства K пространство Крылова $K_m(\mathbf{r}_0, \mathbf{A})$, а в качестве пространства L – $L_m(\tilde{\mathbf{r}}_0, \mathbf{A}^T)$, где вектор $\tilde{\mathbf{r}}_0$ удовлетворяет условию $(\mathbf{r}_0, \tilde{\mathbf{r}}_0) \neq 0$. Тогда, если в методе

FOM заменить процедуру Арнольди процедурой Ланцоша, решение будет уточняться по формуле

$$\mathbf{x}_m = \mathbf{x}_0 + \beta \mathbf{V}_m \mathbf{T}_m^{-1} \mathbf{e}_1 , \tag{3.63}$$

где $\beta = (\mathbf{r}_0,\ \tilde{\mathbf{r}}_0)$. Решение СЛАУ с трехдиагональной матрицей, очевидно, удобнее всего искать с помощью метода прогонки (см. 2.15). Описанная модификация FOM носит название двойственного метода Ланцоша. Однако подобная схема все еще требует одновременного хранения всех базисных векторов либо их перевычисления после нахождения коэффициентов. Поэтому на практике обычно используется другой подход.

Запишем LU-разложение для матрицы $\mathbf{T}_m$

$$\mathbf{T}_m = \mathbf{L}_m \mathbf{U}_m,$$

и определим матрицу $\mathbf{P}_m$ следующим образом:

$$\mathbf{P}_m = \mathbf{V}_m \mathbf{U}_m^{-1} . \tag{3.64}$$

Подставим это выражение в (3.33)

$$\mathbf{x}_m = \mathbf{x}_0 + \beta \mathbf{V}_m \mathbf{U}_m^{-1} \mathbf{L}_m^{-1} \mathbf{e}_1 = \mathbf{x}_0 + \beta \mathbf{P}_m \mathbf{L}_m^{-1} \mathbf{e}_1. \tag{3.65}$$

Рассмотрим теперь свойства матрицы $\mathbf{P}_m$. По аналогии с (3.64) запишем

$$\tilde{\mathbf{P}}_m = \mathbf{W}_m (\mathbf{L}_m^T)^{-1} . \tag{3.66}$$

Тогда имеет место

$$(\tilde{\mathbf{P}}_m)^T \mathbf{A} \mathbf{P}_m = \mathbf{L}_m^{-1} \mathbf{W}_m^T \mathbf{A} \mathbf{V}_m \mathbf{U}_m^{-1} = \mathbf{L}_m^{-1} \mathbf{T}_m \mathbf{U}_m^{-1} = \mathbf{E}_m \mathbf{D}_m , \tag{3.67}$$

где $\mathbf{D}_m$ есть диагональная матрица с элементами

$$d_{ii} = (\mathbf{v}_i, \mathbf{w}_i).$$

Если обозначить образующие $\mathbf{P}_m$ столбцы векторами $\mathbf{p}_k$, а столбцы $\tilde{\mathbf{P}}_m$ – векторами $\tilde{\mathbf{p}}_k$, $k = 1, \ldots, m$, то (2.59) означает, что

$$i \neq j \Rightarrow \tilde{\mathbf{p}}_i^T \mathbf{A} \mathbf{p}_j = (\mathbf{A} \mathbf{p}_i,\ \tilde{\mathbf{p}}_j) = 0 . \tag{3.68}$$

Системы векторов $\{\tilde{\mathbf{p}}_i\}_{i=1}^m$ и $\{\mathbf{p}_i\}_{i=1}^m$, удовлетворяющие условию (3.68), называются A-бисопряженными или, если не возникает неоднозначности, просто бисопряженными.

Очевидно, что бисопряженность эквивалентна биортогональности скалярного A-произведения, а потому для нахождения векторов $\tilde{\mathbf{p}}$ и $\mathbf{p}$ вместо соотношений (3.64) и (3.66) можно пользоваться биортогонализацией Ланцоша (3.33) – (3.36) с использованием в ней вместо скалярных произведений скалярных A-произведений[1].

[1] $(\mathbf{x}, \mathbf{y})_A = (\mathbf{A}\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{A}^T\mathbf{y}) = \mathbf{y}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{A}^T\mathbf{y}$

Такая вычислительная схема носит название метода бисопряженных градиентов или BiCG[1] [51]. В качестве вектора $\tilde{\mathbf{r}}_0$ на практике обычно выбирается $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$ или $\tilde{\mathbf{r}}_0 = \mathbf{e}_k$, где индекс k таков, что $[\mathbf{r}_0]_k \neq 0$. Как и двойственный метод Ланцоша, BiCG может быть легко приспособлен для решения системы $\mathbf{A}^T\mathbf{z} = \tilde{\mathbf{b}}$ одновременно с $\mathbf{Ax} = \mathbf{b}$.

Алгоритм метода BiCG

Построить матрицу предобусловливателя M

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

Выбрать вектор $\tilde{\mathbf{r}}_0$, **удовлетворяющий условию** $(\mathbf{r}_0, \tilde{\mathbf{r}}_0) \neq 0$ (например, $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$)

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

Найти $\mathbf{z}_{i-1}$ **из системы** $\mathbf{M}\,\mathbf{z}_{i-1} = \mathbf{r}_{i-1}$

Найти $\tilde{\mathbf{z}}_{i-1}$ **из системы** $\mathbf{M}^T\,\tilde{\mathbf{z}}_{i-1} = \tilde{\mathbf{r}}_{i-1}$

$\rho_{i-1} = (\mathbf{z}_{i-1}, \tilde{\mathbf{r}}_{i-1})$

Если $\rho_{i-1} = 0$

то метод не может решить данную систему

Если $i = 1$

$\mathbf{p}_i = \mathbf{z}_i$

$\tilde{\mathbf{p}}_i = \tilde{\mathbf{z}}_{i-1}$

Иначе

$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$

$\mathbf{p}_i = \mathbf{z}_i + \beta_{i-1}\mathbf{p}_{i-1}$

$\tilde{\mathbf{p}}_i = \tilde{\mathbf{z}}_{i-1} + \beta_{i-1}\tilde{\mathbf{p}}_{i-1}$

$\mathbf{q}_i = \mathbf{A}\,\mathbf{p}_i$

$\tilde{\mathbf{q}}_i = \mathbf{A}^T\tilde{\mathbf{p}}_i$

$\alpha_i = \rho_{i-1}/(\tilde{\mathbf{p}}_i,\ \mathbf{q}_i)$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\,\mathbf{p}_i$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i\,\mathbf{q}_i$

$\tilde{\mathbf{r}}_i = \tilde{\mathbf{r}}_{i-1} - \alpha_i\tilde{\mathbf{q}}_i$

Если $\|\mathbf{r}_i\|_2 / \|\mathbf{r}_0\|_2 \leq Tol$

то КОНЕЦ ($\mathbf{x}_i$ **– полученное решение)**

увеличить i

[1] Biconjugate Gradients. (Иногда также употребляется аббревиатура BCG.)

3.4.4. *Свободный от транспонирования метод квази-минимальных невязок*

В 3.4.3 был описан метод бисопряженных градиентов BiCG, имеющий по сравнению с FOM и GMRES гораздо более простую вычислительную схему и меньшие требования к памяти. Однако BiCG (как и любой другой метод, использующий операции с транспонированной матрицей **A**) плохо поддается реализации на многопроцессорных вычислительных системах с распределенной памятью. Все более широкое применение таких систем привело к разработке целого класса методов, в которых операция транспонирования не используется.

Алгебраически это может быть достигнуто за счет изменения специальным образом полинома p_m, которому удовлетворяет последовательность невязок в методах, использующих подпространства Крылова, $r_m = p_m(\mathbf{A})\mathbf{r}_0$.

Так, на использовании вместо $p_m(\mathbf{A})$ полинома $p_m^2(\mathbf{A})$ основан квадратичный метод бисопряженных градиентов (CGS). Стабилизированный метод бисопряженных градиентов (BiCGStab) использует соотношение $r_m = p_m(\mathbf{A})q_m(\mathbf{A})\mathbf{r}_0$, где q_m – специальным образом строящийся полином, такой, что произведение $p_m q_m$ не содержит нечетных степеней. Построение подобных методов, как правило, оказывается значительно более сложным.

Из числа методов, свободных от транспонирования, в настоящее время широко применяется, так называемый, свободный от транспонирования метод квази-минимальных невязок (TFQMR[1]) [52], алгоритм которого приведен ниже.

Алгоритм метода TFQMR

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \tilde{\mathbf{r}}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

$\mathbf{q}_0 = \mathbf{p}_{-1} = \mathbf{d}_0 = 0$

$\gamma_0 = \eta_0 = 0$

$\rho_{-1} = 1,\ \tau = \tau_0 = \|\mathbf{r}_0\|_2$

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

$\rho_i = (\tilde{\mathbf{r}}_0,\ \mathbf{r}_0),\ \beta_i = \rho_i / \rho_{i-1},\ \mathbf{u}_i = \mathbf{r}_i + \beta_i \mathbf{q}_i$

$\mathbf{p}_i = \mathbf{u}_i + \beta_i(\mathbf{q}_i + \beta_i \mathbf{p}_{i-1}),\ \mathbf{v}_i = \mathbf{A}\,\mathbf{p}_i,\ \sigma_i = (\tilde{\mathbf{r}}_0,\ \mathbf{v}_i)$

$\alpha_i = \rho_i / \sigma_i$, $\mathbf{q}_{i+1} = \mathbf{u}_i - \alpha_i \mathbf{v}_i$, $\mathbf{v}_i = \alpha_i(\mathbf{u}_i + \mathbf{q}_{i+1})$

$\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{A}\mathbf{v}_i$

[1] Transpose-Free Quasi-Minimal Residuals.

Если $\|r_{i+1}\|_2 / \tau \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ **– полученное решение)**

Для $m = 2i + 1$ до $2i + 2$

Если $(m + 1)$ четно

То $\omega_{m+1} = \sqrt{\| \mathbf{r}_{i+1} \|_2 \| \mathbf{r}_i \|_2}$

Иначе $\omega_{m+1} = \|\mathbf{r}_{i+1}\|_2$

$$\gamma_m = \frac{\omega_{m+1}}{\tau_{m-1}}, \; c_m = \frac{1}{\sqrt{1+\gamma_m^2}}$$

$$\tau_m = \tau_{m-1}\gamma_m c_m, \; \eta_m = c_m^2 \alpha_i$$

Если m четно

То $\mathbf{y}_m = \mathbf{q}_i$

Иначе $\mathbf{y}_m = \mathbf{u}_i$

$$\mathbf{d}_m = \mathbf{y}_m + \gamma_{m-1}^2 \frac{\eta_{m-1}}{\alpha_i} \mathbf{d}_{m-1}$$

$$\mathbf{x}_m = \mathbf{x}_{m-1} + \eta_m \mathbf{d}_m$$

увеличить m

увеличить i

3.4.5. *Стабилизированный метод бисопряженных градиентов*

В дополнение к уже сказанному, отметим, что, к сожалению, описанный алгоритм BiCG зачастую обнаруживает в экспериментах неустойчивость и осциллирующее поведение нормы невязки. Более того, итерационный процесс может полностью оборваться, без возможности его дальнейшего продления. Это происходит если коэффициент $\beta_j = 0$. Разработанный стабилизированный метод бисопряженных градиентов направлен, в том числе, и на устранение подобных ситуаций [53]. Далее приведен алгоритм стабилизированного метода бисопряженных градиентов.

Алгоритм метода BiCGStab

Построить матрицу предобусловливателя M

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

Выбрать вектор $\tilde{\mathbf{r}}$ **, удовлетворяющий условию** $(\mathbf{r}_0, \tilde{\mathbf{r}}) \neq 0$

(например, $\mathbf{r}_0 = \tilde{\mathbf{r}}$)

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

$\rho_{i-1} = (\tilde{\mathbf{r}}, \mathbf{r}_{i-1})$

Если $\rho_{i-1} = 0$

то метод не может решить данную систему

Если $i = 1$

$\mathbf{p}_i = \mathbf{r}_{i-1}$

Иначе

$\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})\ (\alpha_{i-1} / \omega_{i-1})$

$\mathbf{p}_i = \mathbf{r}_{i-1} + \beta_{i-1}(\mathbf{p}_{i-1} - \omega_{i-1}\, \mathbf{v}_{i-1})$

Найти $\tilde{\mathbf{p}}$ **из системы** $\mathbf{M}\,\tilde{\mathbf{p}} = \mathbf{p}_i$

$\mathbf{v}_i = \mathbf{A}\,\tilde{\mathbf{p}}$

$\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}},\ \mathbf{v}_i)$

$\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\, \mathbf{v}_i$

Если $\|\mathbf{s}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\, \tilde{\mathbf{p}}$ **– полученное решение**)

Найти $\tilde{\mathbf{s}}$ **из системы** $\mathbf{M}\,\tilde{\mathbf{s}} = \mathbf{s}$

$\mathbf{t} = \mathbf{A}\,\tilde{\mathbf{s}}$

$\omega_i = (\mathbf{t}, \mathbf{s}) / (\mathbf{t}, \mathbf{t})$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\, \tilde{\mathbf{p}} + \omega_i\, \tilde{\mathbf{s}}$

$\mathbf{r}_i = \mathbf{s} - \omega_i\, \mathbf{t}$

Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}^{(i)}$ **– полученное решение**)

увеличить i

3.4.6. *Метод квази-минимальных невязок*

Разработка еще одного итерационного метода была направлена на устранение, упомянутой в 3.4.5, проблемы метода BiCG, а именно нестабильной сходимости. Таким методом является метод квази-минимальных невязок (QMR) [54]. Далее приведен алгоритм метода QMR.

Алгоритм метода QMR

Построить матрицу предобусловливателя M

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

$\tilde{\mathbf{v}}_1 = \mathbf{r}_0$

Найти y из системы $\mathbf{M}_1\mathbf{y} = \tilde{\mathbf{v}}_1$

$\rho_1 = \|\mathbf{y}\|_2$

$\mathbf{M}_2^T\mathbf{z} = \tilde{\boldsymbol{\omega}}_1$

$\xi_1 = \|\mathbf{z}\|_2$

$\gamma_0 = 1;\ \eta_0 = -1$

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

Если $\rho_i = 0$ или $\xi_i = 0$

то метод не может решить данную систему

$\mathbf{v}_i = \tilde{\mathbf{v}}_i / \rho_i;\ \mathbf{y} = \mathbf{y} / \rho_i$

$\boldsymbol{\omega}_i = \tilde{\boldsymbol{\omega}}_i / \xi_i;\ \mathbf{z} = \mathbf{z} / \xi_i$

$\delta_i = (\mathbf{z}, \mathbf{y})$

Если $\delta_i = 0$

то метод не может решить данную систему

Найти $\tilde{\mathbf{y}}$ **из системы** $\mathbf{M}_2 \tilde{\mathbf{y}} = \mathbf{y}$

Найти $\tilde{\mathbf{z}}$ **из системы** $\mathbf{M}_1^T \tilde{\mathbf{z}} = \mathbf{z}$

Если $i = 1$

$\mathbf{p}_1 = \tilde{\mathbf{y}}\ ;\ \mathbf{q}_1 = \tilde{\mathbf{z}}$

Иначе

$\mathbf{p}_i = \tilde{\mathbf{y}} - (\xi_i \delta_i / \chi_{i-1}) \mathbf{p}_{i-1}$

$\mathbf{q}_i = \tilde{\mathbf{z}} - (\rho_i \delta_i / \chi_{i-1}) \mathbf{q}_{i-1}$

$\tilde{\mathbf{p}} = \mathbf{A}\, \mathbf{p}_i$

$\chi_i = (\mathbf{q}_i,\ \tilde{\mathbf{p}}\ )$

Если $\chi_i = 0$

то метод не может решить данную систему

$\beta_i = \chi_i / \delta_i$

Если $\beta_i = 0$

то метод не может решить данную систему

$\tilde{\mathbf{v}}_{i+1} = \tilde{\mathbf{p}} - \beta_i \mathbf{v}_i$

Найти y из системы $\mathbf{M}_1 \mathbf{y} = \tilde{\mathbf{v}}_{i+1}$

$\rho_{i+1} = \|\mathbf{y}\|_2$

$\tilde{\boldsymbol{\omega}}_{i-1} = \mathbf{A}^T \mathbf{q}_i - \beta_i \boldsymbol{\omega}_i$

Найти z из системы $\mathbf{M}_2^T \mathbf{z} = \tilde{\boldsymbol{\omega}}_{i-1}$

$\xi_{i+1} = \|\mathbf{z}\|_2$

$\theta_i = \rho_{i+1} / (\gamma_i |\beta_i|)$

$\gamma_i = 1 / \sqrt{1 + \theta_i^2}$

Если $\gamma_i = 0$

то метод не может решить данную систему

$\eta_i = -\eta_{i-1}\rho_i\gamma_i^2 / (\beta_i\gamma_{i-1}^2)$

Если $i = 1$

$\mathbf{d}_1 = \eta_1 \mathbf{p}_1$; $\mathbf{s}_1 = \eta_1 \tilde{\mathbf{p}}$

Иначе

$\mathbf{d}_i = \eta_i \mathbf{p}_i + (\theta_{i-1}\gamma_i)^2 \mathbf{d}_{i-1}$

$\mathbf{s}_i = \eta_i \tilde{\mathbf{p}} + (\theta_{i-1}\gamma_i)^2 \mathbf{s}_{i-1}$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{d}_i$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \mathbf{s}_i$

Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ **– полученное решение**)

увеличить i

Метод характеризуется зачастую более гладкой сходимостью, чем метод бисопряженных градиентов.

3.4.7. *Квадратичный метод сопряженных градиентов*

Еще одним итерационным методом решения СЛАУ является, уже упомянутый, квадратичный метод сопряженных градиентов CGS [55]. Часто он характеризуется повышением скорости решения в два раза по сравнению с методом BiCG. Далее приведен алгоритм метода CGS.

Алгоритм метода CGS

Построить матрицу предобусловливателя M

Выбрать начальное приближение $\mathbf{x}_0$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

Выбрать вектор $\tilde{\mathbf{r}}$, **удовлетворяющий условию** $(\mathbf{r}_0, \tilde{\mathbf{r}}) \neq 0$ (например, $\tilde{\mathbf{r}} = \mathbf{r}_0$)

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

$\rho_{i-1} = (\tilde{\mathbf{r}}, \mathbf{r}_{i-1})$

Если $\rho_{i-1} = 0$

то метод не может решить данную систему

Если $i = 1$

$\mathbf{u}_1 = \mathbf{r}_0$

$\mathbf{p}_1 = \mathbf{u}_1$

Иначе

$\beta_{i-1} = (\rho_{i-1} / \rho_{i-2})$

$\mathbf{u}_i = \mathbf{r}_{i-1} + \beta_{i-1}\mathbf{q}_{i-1}$

$\mathbf{p}_i = \mathbf{u}_i + \beta_{i-1}(\mathbf{q}_{i-1} + \beta_{i-1}\mathbf{p}_{i-1})$

Найти $\tilde{\mathbf{p}}$ **из системы** $\mathbf{M}\tilde{\mathbf{p}} = \mathbf{p}^{(i)}$

$\tilde{\mathbf{v}} = \mathbf{A}\tilde{\mathbf{p}}$

$\alpha_i = \rho_{i-1} / (\tilde{\mathbf{r}}, \tilde{\mathbf{v}})$

$\mathbf{q}_i = \mathbf{u}_i - \alpha_i \tilde{\mathbf{v}}$

Найти $\tilde{\mathbf{u}}$ **из системы** $\mathbf{M}\tilde{\mathbf{u}} = \mathbf{u}_i + \mathbf{q}_i$

$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \tilde{\mathbf{u}}$

$\tilde{\mathbf{q}} = \mathbf{A}\tilde{\mathbf{u}}$

$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i \tilde{\mathbf{q}}$

Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ – **полученное решение**)

увеличить i

3.4.8. *Симметричный случай*

Все ранее описанные итерационные методы при построении не опирались на какие-либо предположения о симметричности матрицы **A**, а поэтому без специальных изменений могут быть применены к симметричным СЛАУ. Однако свойство симметрии матрицы позволяет упростить вычислительную схему, что ведет к меньшим требованиям к вычислительным ресурсам при программной реализации. Как будет показано ниже, основные упрощения могут быть внесены в процедуру построения базиса пространства Крылова.

3.4.8.1. Метод Ланцоша

Заметим, что при симметричности матрицы **A** пространства Крылова $K_m(\boldsymbol{\nu}_1, \mathbf{A})$ и $K_m(\boldsymbol{\omega}_1, \mathbf{A}^T)$ совпадают, если $\boldsymbol{\nu}_1 = \boldsymbol{\omega}_1$. В этом случае биортогонализация Ланцоша приведет к построению одинаковых систем векторов $\{\boldsymbol{\nu}_i\}_{i=1}^m$, $\{\boldsymbol{\omega}_i\}_{i=1}^m$. Условие биортогональности тогда сведется к

$$i \ne j \Rightarrow (\boldsymbol{\nu}_i, \boldsymbol{\omega}_i) = (\boldsymbol{\omega}_i, \boldsymbol{\omega}_i) = 0, \tag{3.69}$$

т.е. будет построен ортогональный базис пространства Крылова.

Сравнивая формулы ортогонализации Арнольди из 3.2.5 и формулы (3.33) – (3.36), нетрудно заметить, что они совпадают с тем лишь отличием что в (3.33) – (3.34) векторы не нормируются. Таким образом, в симметричном случае ортогонализация Арнольди сводится к соотношению

$$\beta_{j+1}\boldsymbol{\nu}_{j+1} = \mathbf{A}\boldsymbol{\nu}_j - \alpha_j\boldsymbol{\nu}_j - \beta_j\boldsymbol{\nu}_{j-1}, \tag{3.70}$$

называемому ортогонализацией Ланцоша. Коэффициенты α_j, β_j образуют трехдиагональную матрицу $\mathbf{T}_m$, для которой справедливо соотношение

$$\mathbf{V}_m^T\mathbf{A}\,\mathbf{V}_m = \mathbf{T}_m. \tag{3.71}$$

Этот же факт можно получить и другим способом. Воспользуемся соотношением (3.31). Если $\mathbf{A} = \mathbf{A}^T$, то $\mathbf{V}_m^T\mathbf{A}\,\mathbf{V}_m = \mathbf{V}_m^T\mathbf{A}^T\mathbf{V}_m = (\mathbf{V}_m^T\mathbf{A}\,\mathbf{V}_m)^T$, тогда

$$\mathbf{H}_m^T = (\mathbf{V}_m^T\mathbf{A}\,\mathbf{V}_m)^T = \mathbf{V}_m^T\mathbf{A}\,\mathbf{V}_m = \mathbf{H}_m .$$

Очевидно, что матрица в форме Хессенберга может быть симметричной, когда она является трехдиагональной.

Рассмотренный в 3.4.1 метод FOM может быть упрощен для симметричного случая, если процедуру Арнольди в нем заменить на (3.70). Для решения трехдиагональной СЛАУ при этом наиболее естественно использовать метод прогонки (см. 2.15). Такая вычислительная схема, называемая методом Ланцоша, приведена далее.

Алгоритм метода Ланцоша

Выбрать начальное приближение $\mathbf{x}_0$

Вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$

$\beta = \|\mathbf{r}_0\|_2$, $\boldsymbol{\nu}_1 = \mathbf{r}_0 / \beta$

$\beta_1 = 0$, $\boldsymbol{\nu}_1 = 0$

Для $j = 1, \ldots, m$

$\boldsymbol{\omega}_j = \mathbf{A}\boldsymbol{\nu}_j - \beta_j\boldsymbol{\nu}_{j-1}$

$\alpha_j = (\boldsymbol{\omega}_j, \boldsymbol{\nu}_j)$

$\beta_{j+1} = \|\boldsymbol{\omega}_j\|_2$

Если $\beta_{j+1} = 0$

То $m = j$

Прервать цикл по j

Иначе

$\boldsymbol{\nu}_{j+1} = \boldsymbol{\omega}_j / \beta_{j+1}$

Увеличить j

$\mathbf{T}_m = \mathrm{tridiag}(\beta_i, \alpha_j, \beta_{i+1})$, $i = 1, \ldots, m$

Найти y **из трехдиагональной системы** $\mathbf{T}_m\mathbf{y} = \beta\mathbf{e}_1$

$$\mathbf{x}_m = \mathbf{x}_0 + \sum_{i=1}^{m} y_i\mathbf{v}_i$$

3.4.8.2. Метод сопряженных градиентов

Как уже отмечалось, в случае симметричной матрицы пространства Крылова $K_m(\mathbf{r}_0, \mathbf{A})$ и $K_m(\mathbf{r}_0, \mathbf{A}^T)$ совпадают. Это приводит к тому, что в методе бисопряженных градиентов будут иметь место равенства

$$\tilde{\mathbf{r}}_i \equiv \mathbf{r}_i,\ \tilde{\mathbf{p}}_i \equiv \mathbf{p}_i$$

при выборе $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$.

Очевидное упрощение вычислительной схемы сводится к тому, что отпадает необходимость в хранении и обработке векторов $\tilde{\mathbf{r}}_i$ и $\tilde{\mathbf{p}}_i$. Соотношение (3.68) при этом превращается в условие

$$i \neq j \Rightarrow \mathbf{p}_i^T \mathbf{A}\, \mathbf{p}_j = 0 . \qquad (3.72)$$

Векторы $\{\mathbf{p}_i\}_{i=1}^m$, удовлетворяющие условию (3.72), называются A-сопряженными или, если не возникает неоднозначности, просто сопряженными.

Если в дополнение к симметричности потребовать, чтобы матрица $\mathbf{A}$ была положительно определенной, то при $\mathbf{p}_i \neq 0$ всегда $\mathbf{p}_i^T \mathbf{A}\, \mathbf{p}_i \neq 0$ и сходимость метода гарантирована. Метод, который получается за счет упрощений, вносимых симметричностью, называется методом сопряженных градиентов (CG[1]) [56]. При этом CG относится к классу тех проекционных методов, для которых $K = L(= K_m(\mathbf{r}_0, \mathbf{A}))$.

Алгоритм метода CG

Построить матрицу предобусловливателя M
Выбрать начальное приближение $\mathbf{x}_0$
Вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\, \mathbf{x}_0$
Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}
    Найти $\mathbf{z}_{i-1}$ **из системы** $\mathbf{M}\, \mathbf{z}_{i-1} = \mathbf{r}_{i-1}$
    $\rho_{i-1} = (\mathbf{r}_{i-1}, \mathbf{z}_{i-1})$
    Если $i = 1$
        $\mathbf{p}_1 = \mathbf{z}_0$
    Иначе
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $\mathbf{p}_i = \mathbf{z}_{i-1} + \beta_{i-1}\, \mathbf{p}_{i-1}$
    $\mathbf{q}_i = \mathbf{A}\, \mathbf{p}_i$
    $\alpha_i = \rho_{i-1} / (\mathbf{p}_i, \mathbf{q}_i)$
    $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\, \mathbf{p}_i$
    $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i\, \mathbf{q}_i$
    Если $\|\mathbf{r}\|_2 / \|\mathbf{r}_0\|_2 \leq Tol$
        то КОНЕЦ ($\mathbf{x}_i$ **– полученное решение**)
увеличить i

Следует заметить, что метод бисопряженных градиентов исторически появился как обобщение CG на несимметричный случай, что и отрази-

[1] Conjugate Gradient Method

лось на его названии, так как вместо ортогонализации (3.70) в нем используется биортогонализация (3.33)–(3.36). Приведем алгоритм метода сопряженных градиентов.

3.4.9. *О других итерационных методах*

Применение метода сопряженных градиентов к системе

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}, \qquad (3.73)$$

дает метод, носящий название метод сопряженных градиентов после первой трансформации Гаусса [57]. В иностранной литературе этот метод принято называть CGNR[1]. Далее приведен его алгоритм.

Алгоритм метода CGNR

Выбрать начальное приближение $\mathbf{x}_0$

Вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$, $\mathbf{z}_0 = \mathbf{A}^T\mathbf{r}_0$, $\mathbf{p}_0 = \mathbf{z}_0$

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

$\mathbf{w}_i = \mathbf{A}\mathbf{p}_i$

$\alpha_i = \| \mathbf{z}_i \|_2^2 / \| \mathbf{w}_i \|_2^2$

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i\,\mathbf{p}_i$

$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i\,\mathbf{w}_i$

$\mathbf{z}_{i+1} = \mathbf{A}^T\mathbf{r}_{i+1}$

$\beta_i = \| \mathbf{z}_{i+1} \|_2^2 / \| \mathbf{z}_i \|_2^2$

$\mathbf{p}_{i+1} = \mathbf{z}_{i+1} + \beta_i\,\mathbf{p}_i$

Если $\|\mathbf{r}_{i+1}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ – **полученное решение**)

увеличить i

Другую вычислительную схему получают применением метода сопряженных градиентов к системе

$$\mathbf{A}\mathbf{A}^T\mathbf{y} = \mathbf{b},\ \mathbf{x} = \mathbf{A}^T\mathbf{y}. \qquad (3.74)$$

Такую вычислительную схему принято называть – метод сопряженных градиентов после второй трансформации Гаусса [57]. В иностранной литературе этот метод носит название CGNE[2]. Приведем его алгоритм.

Алгоритм метода CGNE

Выбрать начальное приближение $\mathbf{x}_0$

Вычислить $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$, $\mathbf{p}_0 = \mathbf{A}^T\mathbf{r}_0$

Для $i = 1, 2, \ldots$ **до сходимости или до** N_{it}^{max}

$\alpha_i = (\mathbf{r}_i, \mathbf{r}_i) / (\mathbf{p}_i, \mathbf{p}_i)$

[1] Conjugate Gradient Normal equation Residual method.
[2] Conjugate Gradient Normal Equation method.

$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$

$\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A}\mathbf{p}_i$

$\beta_i = (\mathbf{r}_i, \mathbf{r}_i) / (\mathbf{p}_i, \mathbf{p}_i)$

$\mathbf{p}_{i+1} = \mathbf{A}^T\mathbf{r}_{i+1} + \beta_i \mathbf{p}_i$

Если $\|\mathbf{r}_{i+1}\|_2 / \|\mathbf{r}_0\|_2 \le Tol$

то КОНЕЦ ($\mathbf{x}_i$ **– полученное решение**)

увеличить i

Методы CGNR и CGNE можно использовать для решения несимметричных СЛАУ. Однако их редко используют на практике, поскольку число обусловленности систем (3.73) и (3.74) равно квадрату числа обусловленности исходной системы $\mathbf{Ax} = \mathbf{b}$.

Существует большое количество и других итерационных методов, о которых можно узнать, например, из [22, 57, 58].

4. ИСПОЛЬЗОВАНИЕ ИТЕРАЦИОННЫХ МЕТОДОВ ПРИ РЕШЕНИИ СЛАУ С ПЛОТНОЙ МАТРИЦЕЙ В АНАЛИЗЕ ПРОВОДНЫХ АНТЕНН

Особая необходимость в решении СЛАУ возникает при использовании широкого класса моделей и подходов, используемых при автоматизированном проектировании аппаратуры. Так, например, решение задач излучения или рассеяния электромагнитной волны сложными объектами может быть получено с помощью интегральных уравнений, сводящихся методом моментов к СЛАУ с плотными, комплексными и несимметричными матрицами. Вычислительные трудности их решения во многом обусловлены заполненностью матриц, приводящей к огромному объему вычислений (особенно в трехмерных задачах).

Как было показано выше, для использования итерационных методов необходимо для каждого метода задать параметры для вычисления. В общем случае каждый набор будет приводить к разным результатам (прежде всего, по времени решения СЛАУ). Так, например, заданный набор значений параметров методов может привести к очень быстрой сходимости для одного из итерационных методов и к посредственной сходимости для другого. Таким образом, выбор значений параметров итерационных методов является достаточно сложной задачей.

Далее приведены некоторые примеры использования итерационных методов (в большинстве случаев BiCGStab). Как правило, результаты использования метода сопровождаются сравнением с результатами, полученными методом исключения Гаусса без выбора ведущего элемента.

4.1. Сравнение итерационных методов без использования предобусловливания

Необходимость использования предобусловливания при итерационном решении проиллюстрируем на примере определения тока в проводной антенне. Все вычисления производились в системе компьютерного моделирования сложных структур проводников и диэлектриков TALGAT [59]. В качестве тестовых конфигураций взяты четыре антенны.

Пример 1. Трапециевидная зубчатая антенна [60]. Параметры матрицы: $N = 765$, плотность dA = 99,997%.

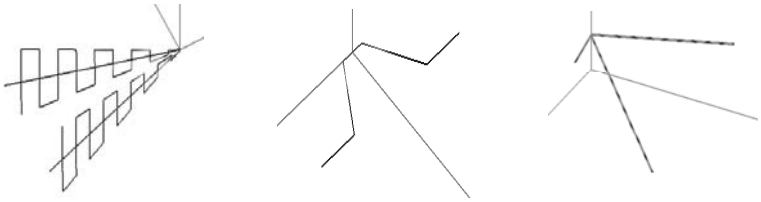
Пример 2. Антенна «чайка» [60] ($N = 1865$, плотность dA = 100%).

Пример 3. Широкодиапазонная антенна [61] ($N = 1261$, dA = 100%).

Пример 4. Диполь с углом между лучами 180° ($N = 2335$, dA = 100%).

Общий вид этих антенн (кроме диполя) в системе TALGAT приведен на рис. 4.1, а результаты использования (T – время решения СЛАУ, N_{it} –

число итераций) итерационных методов без предобусловливания показаны в табл. 4.1. За начальное приближение вектора решения принималось равенство всех его элементов 0.1. Итерации останавливались при условии $\|\mathbf{r}_k\|_2 / \|\mathbf{r}_0\|_2 < Tol = 10^{-8}$, где $\mathbf{r}_k$ и $\mathbf{r}_0$ – невязки после k-й итерации и начального приближения, соответственно, или при достижении 400 итераций.



а *б* *в*

Рис. 4.1. Общий вид антенн в системе TALGAT: трапециевидной зубчатой (*а*), «чайка» (*б*), широкодиапазонной (*в*)

Таблица 4.1

Результаты использования итерационных методов без предобусловливания

Номер примера		BiCG	BiCGStab	CGS	QMR	GMRES(m)		
						$m = 10$	$m = 20$	$m = 40$
1	T, с	39	36	33	41	97	83	75
	N_{it}	400	400	400	400	192	86	39
2	T, с	236	225	193	244	1218	2296	4648
	N_{it}	400	400	400	400	400	400	400
3	T, с	93	86	76	94	470	906	1813
	N_{it}	400	400	400	400	400	400	400
4	T, с	297	272	241	301	1471	2831	5592
	N_{it}	400	400	400	400	400	400	400

Из приведенных результатов видно, что применение итерационных методов в классической постановке (без предобусловливания) не вполне выгодно, поскольку в большинстве случаев для получения приемлемого решения итерационному процессу было не достаточно 400 итераций или соответствовало значительным временным затратам (намного большим, чем при использовании метода Гаусса).

4.2. Сравнение итерационных методов при использовании предобусловливания

Первоначально для выбора предпочтительного итерационного метода система $\mathbf{Ax} = \mathbf{b}$ решалась, как и ранее, в разделе 4.1 итерационными методами (BiCG, BiCGStab, CGS, QMR и GMRES(m) при $m = 10, 20, 40$) на

примере определения токов в проводной антенне. В качестве исследуемых структур взяты две конфигурации антенн, использованные как тестовые в разделе 4.1. Первой была выбрана простая конфигурация: диполь с углом между лучами 180° ($N = 2335$, dA = 100%). В качестве второй выступила широкодиапазонная антенна (рис. 4.1 *в*) ($N = 1261$, dA = 100%).

За начальное приближение вектора решения принималось равенство всех его элементов 0.1. Итерационный процесс останавливался при условии $||\mathbf{r}_k||_2/||\mathbf{r}_0||_2 < Tol = 10^{-8}$ или при достижении 180 итераций. В качестве способов формирования матрицы предобусловливания использовались: полное LU-разложение; ILU(0) и ILU(p), при $p = 1, 2, 3$. В качестве способа предфильтрации использовался подход, основанный на нормах строк матрицы (3.55). Вычисления выполнены для 8 значений допуска обнуления τ, начиная с 10^{-2} с уменьшением в два раза. Результаты вычислений (T – время решения СЛАУ, N_{it} – число итераций) приведены в табл. 4.2 для диполя и в табл. 4.3 для широкодиапазонной антенны.

Таблица 4.2

Результаты работы итерационных методов для диполя при $N = 2335$

Метод	τ	LU		ILU(0)		ILU(1)		ILU(2)		ILU(3)	
		T, с	N_{it}	T, с	N_{it}	T, с	N_{it}	T, с	N_{it}	T, с	N_{it}
1	2	3	4	5	6	7	8	9	10	11	12
BiCG	10^{-2}	215	180	215	180	230	180	227	180	227	180
	$5{\cdot}10^{-3}$	109	88	200	167	203	161	170	133	139	107
	10^{-3}	37	25	57	42	60	42	59	41	59	41
	$5{\cdot}10^{-4}$	33	21	49	35	51	34	52	35	51	34
	10^{-4}	25	13	34	21	37	21	37	21	37	21
	$5{\cdot}10^{-5}$	24	11	32	18	35	18	35	18	35	18
	10^{-5}	27	7	31	10	37	12	36	11	31	7
	$5{\cdot}10^{-6}$	38	6	40	6	43	6	43	6	43	6
BiCGStab	10^{-2}	111	107	179	180	168	168	180	180	153	152
	$5{\cdot}10^{-3}$	66	64	129	127	122	120	122	119	136	134
	10^{-3}	28	20	49	43	49	42	48	41	46	39
	$5{\cdot}10^{-4}$	25	17	38	31	42	34	39	31	37	29
	10^{-4}	20	10	25	16	25	15	27	17	26	16
	$5{\cdot}10^{-5}$	17	7	24	13	25	13	25	13	24	12
	10^{-5}	25	5	26	6	29	7	29	7	27	5
	$5{\cdot}10^{-6}$	36	4	38	4	39	4	39	4	39	4
CGS	10^{-2}	165	180	165	180	189	180	189	180	189	180
	$5{\cdot}10^{-3}$	91	95	165	180	178	179	164	166	152	155
	10^{-3}	27	21	42	38	44	38	43	36	43	36
	$5{\cdot}10^{-4}$	23	17	35	30	38	31	38	31	36	29
	10^{-4}	19	10	24	16	25	15	26	16	26	16
	$5{\cdot}10^{-5}$	17	7	23	14	24	13	23	12	23	12
	10^{-5}	23	5	26	7	34	13	201	180	26	5
	$5{\cdot}10^{-6}$	33	3	35	3	38	3	38	3	38	3

Окончание табл. 4.2

1	2	3	4	5	6	7	8	9	10	11	12
QMR	10^{-2}	189	180	189	180	219	180	219	180	219	180
	$5\cdot10^{-3}$	189	180	189	180	219	180	219	180	219	180
	10^{-3}	55	46	189	180	220	180	220	180	220	180
	$5\cdot10^{-4}$	43	34	189	180	222	180	222	180	222	180
	10^{-4}	27	17	193	180	225	180	223	180	223	180
	$5\cdot10^{-5}$	24	13	195	180	227	180	227	180	227	180
	10^{-5}	26	7	212	180	246	180	244	180	30	7
	$5\cdot10^{-6}$	37	6	39	6	43	6	43	6	43	6
GMRES(10)	10^{-2}	369	69	376	72	465	83	430	76	408	72
	$5\cdot10^{-3}$	318	60	385	73	358	64	380	67	351	62
	10^{-3}	87	15	122	22	126	21	116	20	124	21
	$5\cdot10^{-4}$	55	9	92	16	108	18	100	16	96	16
	10^{-4}	27	4	42	7	49	7	55	8	47	7
	$5\cdot10^{-5}$	22	2	40	6	40	5	39	5	38	5
	10^{-5}	24	1	28	2	31	2	30	2	27	1
	$5\cdot10^{-6}$	36	1	38	1	39	1	39	1	39	1
GMRES(20)	10^{-2}	278	27	340	34	391	36	356	33	352	32
	$5\cdot10^{-3}$	228	22	296	29	288	26	301	28	266	24
	10^{-3}	60	6	92	9	99	9	100	9	96	9
	$5\cdot10^{-4}$	47	4	79	7	84	7	78	7	84	7
	10^{-4}	19	1	34	3	40	3	39	3	35	3
	$5\cdot10^{-5}$	19	1	27	2	28	2	27	2	26	2
	10^{-5}	24	1	27	1	29	1	29	1	27	1
	$5\cdot10^{-6}$	36	1	39	1	39	1	39	1	39	1
GMRES(40)	10^{-2}	244	12	282	14	325	15	300	14	300	14
	$5\cdot10^{-3}$	202	10	224	12	225	11	225	11	216	10
	10^{-3}	26	1	80	4	88	4	88	4	88	4
	$5\cdot10^{-4}$	23	1	56	3	59	3	59	3	59	3
	10^{-4}	19	1	25	1	27	1	27	1	27	1
	$5\cdot10^{-5}$	19	1	23	1	25	1	25	1	25	1
	10^{-5}	24	1	27	1	29	1	29	1	27	1
	$5\cdot10^{-6}$	36	1	38	1	40	1	39	1	40	1

Таблица 4.3

Результаты работы итерационных методов для антенны из рис. 4.1 ***в*** **при** $N = 1261$

Метод	τ	LU		ILU(0)		ILU(1)		ILU(2)		ILU(3)	
		T, с	N_{it}	T, с	N_{it}	T, с	N_{it}	T, с	N_{it}	T, с	N_{it}
1	2	3	4	5	6	7	8	9	10	11	12
BiCG	10^{-2}	23	59	26	68	26	66	27	70	25	64
	$5\cdot10^{-3}$	19	46	22	56	21	53	20	51	21	52
	10^{-3}	12	25	13	32	14	32	13	30	14	31
	$5\cdot10^{-4}$	11	20	11	26	11	22	11	24	11	22
	10^{-4}	11	12	10	18	12	16	13	18	13	16
	$5\cdot10^{-5}$	18	10	13	14	18	13	20	15	21	14
	10^{-5}	53	7	37	11	58	10	56	7	56	7
	$5\cdot10^{-6}$	52	5	48	11	56	5	56	5	56	5

Окончание табл. 4.3

1	2	3	4	5	6	7	8	9	10	11	12
BiCGStab	10^{-2}	20	61	20	64	20	65	21	68	22	70
	$5\cdot10^{-3}$	17	50	16	50	17	53	16	49	21	68
	10^{-3}	10	24	10	27	10	26	10	28	10	28
	$5\cdot10^{-4}$	8	16	8	21	8	20	9	21	8	19
	10^{-4}	10	8	8	14	9	12	10	13	10	11
	$5\cdot10^{-5}$	18	7	12	11	15	8	18	10	19	11
	10^{-5}	58	4	39	10	56	8	55	4	55	4
	$5\cdot10^{-6}$	58	3	51	8	54	3	54	3	54	3
CGS	10^{-2}	16	49	20	64	18	59	21	67	17	55
	$5\cdot10^{-3}$	13	38	17	54	15	48	14	42	15	45
	10^{-3}	9	21	9	26	9	25	9	25	10	28
	$5\cdot10^{-4}$	8	15	8	20	8	19	8	20	8	17
	10^{-4}	9	8	8	13	9	12	10	13	10	11
	$5\cdot10^{-5}$	16	7	12	11	16	9	17	10	18	10
	10^{-5}	51	4	37	9	56	8	55	4	55	4
	$5\cdot10^{-6}$	51	3	47	8	54	3	54	3	54	3
QMR	10^{-2}	65	180	65	180	65	180	65	180	65	180
	$5\cdot10^{-3}$	44	118	65	180	37	102	31	84	42	114
	10^{-3}	16	36	29	76	24	61	21	53	22	57
	$5\cdot10^{-4}$	12	25	65	180	65	180	66	180	66	180
	10^{-4}	13	14	69	180	72	180	73	180	74	180
	$5\cdot10^{-5}$	19	10	78	180	86	180	88	180	89	180
	10^{-5}	54	7	119	180	149	180	57	7	57	7
	$5\cdot10^{-6}$	53	5	134	180	56	5	56	5	56	5
GMRES(10)	10^{-2}	27	15	49	29	59	35	70	42	50	29
	$5\cdot10^{-3}$	23	12	52	31	43	25	49	29	52	30
	10^{-3}	11	5	65	39	19	10	27	15	29	16
	$5\cdot10^{-4}$	9	4	24	13	12	6	39	22	13	7
	10^{-4}	10	2	16	7	12	4	16	6	13	4
	$5\cdot10^{-5}$	17	2	13	3	18	3	20	3	21	4
	10^{-5}	51	1	38	2	57	2	55	1	55	1
	$5\cdot10^{-6}$	51	1	49	2	55	1	55	1	55	1
GMRES(20)	10^{-2}	26	8	32	10	40	12	42	13	42	13
	$5\cdot10^{-3}$	17	5	34	10	29	9	29	9	53	16
	10^{-3}	10	3	21	6	13	4	18	5	17	5
	$5\cdot10^{-4}$	8	2	14	4	9	2	15	4	9	2
	10^{-4}	9	1	9	2	9	2	11	2	10	1
	$5\cdot10^{-5}$	16	1	12	1	16	1	17	1	18	1
	10^{-5}	51	1	36	1	56	1	55	1	55	1
	$5\cdot10^{-6}$	51	1	47	1	55	1	54	1	55	1
GMRES(40)	10^{-2}	19	3	25	4	31	5	34	5	33	5
	$5\cdot10^{-3}$	15	2	23	4	20	3	21	3	27	4
	10^{-3}	9	1	12	2	9	1	12	2	11	2
	$5\cdot10^{-4}$	8	1	8	1	8	1	8	1	8	1
	10^{-4}	9	1	8	1	9	1	10	1	10	1
	$5\cdot10^{-5}$	17	1	12	1	16	1	17	1	18	1
	10^{-5}	51	1	36	1	56	1	55	1	55	1
	$5\cdot10^{-6}$	51	1	47	1	54	1	55	1	55	1

Анализ табл. 4.2, 4.3 показывает, что наиболее предпочтительными, с точки зрения минимизации времени решения СЛАУ с плотной матрицей, получающейся при анализе антенн методом моментов, оказались два метода: BiCGStab и CGS при использовании полного LU-разложения и способа предфильтрации (3.55). Поскольку они показали практически одинаковые результаты, для дальнейших исследований был выбран один из них: метод BiCGStab. Также видно, что в качестве способа формирования матрицы предобусловливания предпочтительнее использовать полное LU-разложение, поскольку оно оказалось наиболее работоспособным для каждого метода. Интересно, что для всех методов существует значение τ, минимизирующее время решения СЛАУ. Поэтому важно исследовать это подробнее.

4.3. Оптимизация допуска обнуления при решении СЛАУ итерационным методом BiCGStab с предобусловливанием

Система $\mathbf{Ax} = \mathbf{b}$ решалась методом BiCGStab на примере определения токов в проводной антенне, приведенной на рис. 4.1 *в*, с матрицей $N = 243$ при $\tau = 10^{-1}, 5\cdot10^{-2}, \ldots, 10^{-4}$ [48, 62]. В качестве способа предфильтрации использовался подход (3.55). При формировании матрицы предобусловливания использовалось полное LU-разложение.

Таблица 4.4

Результаты вычислений при $\tau = 10^{-3}$ для проводной антенны с матрицей $N = 243$

Tol	*T*, с	N_{it}	Re(i_1), A	Im(i_1), A	Re(i_{242}), A	Im(i_{242}), A
10^{-1}	0.16	4	$-5.5433315933\cdot10^{-3}$	$8.5280934358\cdot10^{-4}$	$3.1026158901\cdot10^{-3}$	$-2.0675906220\cdot10^{-4}$
10^{-2}	0.22	5	$-5.5429392637\cdot10^{-3}$	$8.5304111642\cdot10^{-4}$	$3.1019734236\cdot10^{-3}$	$-2.0697442283\cdot10^{-4}$
10^{-3}	0.22	5	$-5.5429392637\cdot10^{-3}$	$8.5304111642\cdot10^{-4}$	$3.1019734236\cdot10^{-3}$	$-2.0697442283\cdot10^{-4}$
10^{-4}	0.22	5	$-5.5429392637\cdot10^{-3}$	$8.5304111642\cdot10^{-4}$	$3.1019734236\cdot10^{-3}$	$-2.0697442283\cdot10^{-4}$
10^{-5}	0.27	6	$-5.5429393543\cdot10^{-3}$	$8.5304010642\cdot10^{-4}$	$3.1019740107\cdot10^{-3}$	$-2.0697295482\cdot10^{-4}$
10^{-6}	0.28	7	$-5.5429393483\cdot10^{-3}$	$8.5304010471\cdot10^{-4}$	$3.1019740055\cdot10^{-3}$	$-2.0697295161\cdot10^{-4}$
10^{-7}	0.28	7	$-5.5429393483\cdot10^{-3}$	$8.5304010471\cdot10^{-4}$	$3.1019740055\cdot10^{-3}$	$-2.0697295161\cdot10^{-4}$
10^{-8}	0.33	8	$-5.5429393528\cdot10^{-3}$	$8.5304010530\cdot10^{-4}$	$3.1019740083\cdot10^{-3}$	$-2.0697295185\cdot10^{-4}$
GE	1.37	–	$-5.5429393528\cdot10^{-3}$	$8.5304010530\cdot10^{-4}$	$3.1019740083\cdot10^{-3}$	$-2.0697295185\cdot10^{-4}$

В табл. 4.4 приведена выборка результатов вычислений при $\tau = 10^{-3}$ (T – время решения СЛАУ, N_{it} – число итераций, реальная и мнимая части второго и последнего элементов вектора решения, т.е. распределения тока в антенне) методом BiCGStab при разной точности вычислений *Tol* ($||\mathbf{r}_k||_2 / ||\mathbf{r}_0||_2 < Tol$, где $\mathbf{r}_k$ и $\mathbf{r}_0$ – невязки после k-й итерации и нулевого начального приближения), а также методом Гаусса (последняя строка). Из неё видно, что методом BiCGStab решение, например с точностью до 4 знаков после запятой совпадающее с решением методом Гаусса, получа-

ется всего за 5 итераций и в 6 раз быстрее, чем методом Гаусса. При необходимости, более точные результаты легко получить, увеличивая число итераций: например решение, совпадающее с точностью до 10 знаков, получается за 8 итераций и в 4 раза быстрее. И наоборот, если важна не точность, а скорость вычисления, то её можно увеличить примерно в полтора раза выбором высокого значения *Tol*.

Зависимости времени решения СЛАУ с матрицей $N = 243$ от τ для различных *Tol* приведены на рис. 4.2 *а*. Видно, что для всех *Tol* существует значение τ, оптимальное по критерию минимизации времени решения СЛАУ, и выбор значения $\tau = 10^{-3}$, близкого к оптимальному, позволяет ускорить решение в 2–3 раза по сравнению с τ на краях диапазона.

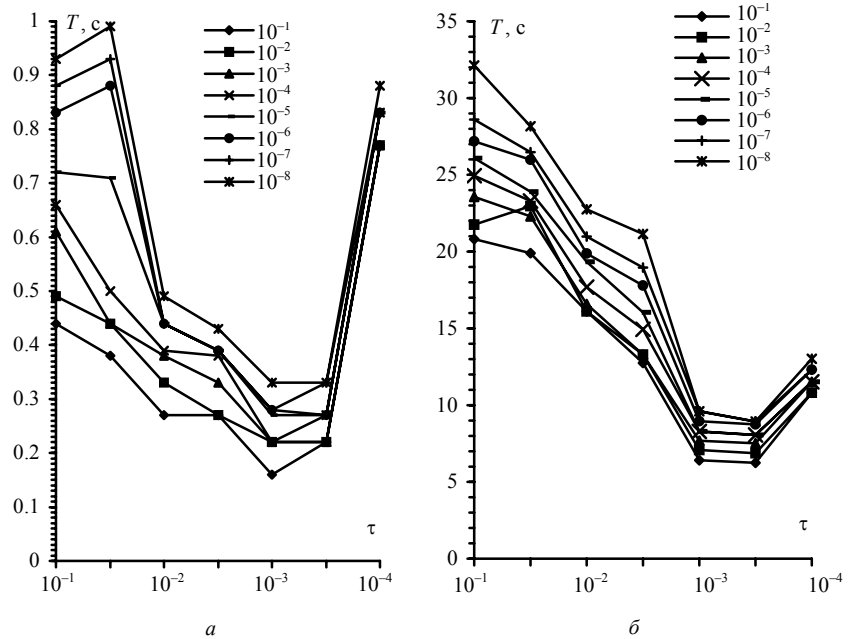


Рис. 4.2. Зависимости времени решения СЛАУ от τ для разных *Tol* при $N = 243$(*а*); 1023(*б*)

Проведены аналогичные вычисления для той же антенны с матрицей $N = 1023$. Выборка результатов вычислений при $\tau = 10^{-3}$ приведена в табл. 4.5. Из неё видно, что методом BiCGStab решение, с точностью до 4 знаков после запятой совпадающее с решением методом Гаусса, получается за 10 итераций и в 14 раз быстрее, чем методом Гаусса. Более точное решение, совпадающее с точностью до 10 знаков, получается за 14 итера-

ций и в 10 раз быстрее. Опять же, скорость вычисления можно увеличить примерно в полтора раза выбором высокого *Tol*.

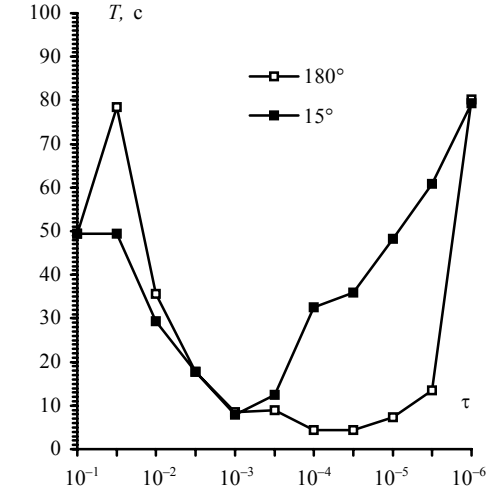
Таблица 4.5

Результаты вычислений при $\tau = 10^{-3}$ для проводной антенны с матрицей $N = 1023$

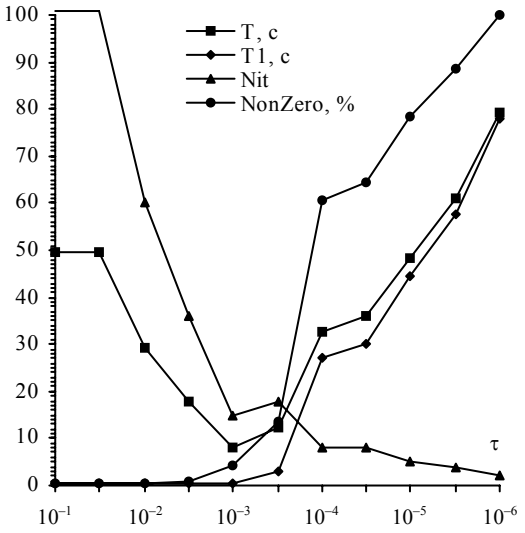
Tol	*T*, с	N_{it}	Re(i_1)	Im (i_1)	Re(i_{1022})	Im(i_{1022})
10^{-1}	6.42	9	$-1.1375442057\cdot10^{-3}$	$2.3606560246\cdot10^{-3}$	$-3.6249794809\cdot10^{-4}$	$1.4353148535\cdot10^{-3}$
10^{-2}	7.09	10	$-1.1371536423\cdot10^{-3}$	$2.3605465706\cdot10^{-3}$	$-3.6221192711\cdot10^{-4}$	$1.4354817163\cdot10^{-3}$
10^{-3}	7.69	11	$-1.1371697921\cdot10^{-3}$	$2.3605427917\cdot10^{-3}$	$-3.6221681167\cdot10^{-4}$	$1.4354648278\cdot10^{-3}$
10^{-4}	8.29	12	$-1.1371694528\cdot10^{-3}$	$2.3605428828\cdot10^{-3}$	$-3.6221694059\cdot10^{-4}$	$1.4354651706\cdot10^{-3}$
10^{-5}	8.29	12	$-1.1371694528\cdot10^{-3}$	$2.3605428828\cdot10^{-3}$	$-3.6221694059\cdot10^{-4}$	$1.4354651706\cdot10^{-3}$
10^{-6}	8.96	13	$-1.1371694505\cdot10^{-3}$	$2.3605428747\cdot10^{-3}$	$-3.6221693789\cdot10^{-4}$	$1.4354651790\cdot10^{-3}$
10^{-7}	9.61	14	$-1.1371694503\cdot10^{-3}$	$2.3605428746\cdot10^{-3}$	$-3.6221693767\cdot10^{-4}$	$1.4354651789\cdot10^{-3}$
10^{-8}	9.61	14	$-1.1371694503\cdot10^{-3}$	$2.3605428746\cdot10^{-3}$	$-3.6221693767\cdot10^{-4}$	$1.4354651789\cdot10^{-3}$
GE	98.92	–	$-1.1371694503\cdot10^{-3}$	$2.3605428746\cdot10^{-3}$	$-3.6221693766\cdot10^{-4}$	$1.4354651789\cdot10^{-3}$

Зависимости времени решения СЛАУ с матрицей $N = 1023$ от τ для различных *Tol* приведены на рис. 4.2 *б*. Видно, что, как и на рис. 4.2 *а*, для всех *Tol* существует оптимальное значение τ. Выбор значения $\tau = 5\cdot10^{-4}$, близкого к оптимальному, позволяет ускорить решение более чем в 3 раза по сравнению с τ на краях диапазона.

Для выяснения зависимости оптимального значения τ от матрицы исследовались (в более широком диапазоне τ) более простые структуры, в частности, диполь с углом между лучами 180° и диполь с углом между лучами 15°, отличающийся характером изменения значений элементов матрицы с удалением от диагонали. Зависимости времени решения СЛАУ (*T*, с) с матрицей $N = 969$ от τ для этих диполей при $Tol = 10^{-8}$ показаны на рис. 4.3 *а*. Видно, что существует оптимальное значение τ и для этих структур. Выбор значений τ, близких к оптимальному, позволяет ускорить решение для угла 180° в 20 раз, а для угла 15° в 10 раз по сравнению с τ на краях диапазона. (На правой границе диапазона τ время решения почти совпадает со временем решения методом Гаусса). Отметим, что в точке $\tau = 10^{-3}$ затраты времени одинаковы для обеих структур, но изменение τ влево и вправо от этой точки даёт изменения времени решения СЛАУ, значительные и противоположные для разных структур. Обращает на себя внимание и то, что рост скорости убывания значений элементов матрицы с удалением от диагонали сдвигает вправо оптимальное значение τ и расширяет (от двух до шести делений, т.е. на два порядка в сторону меньших значений) диапазон значений τ, близких к оптимальному.



а



б

Рис. 4.3. Зависимости характеристик решения СЛАУ от τ при $N = 969$ и $Tol = 10^{-8}$: время решения СЛАУ для угла между лучами диполя 180° и 15° (*а*); характеристики для угла 15° (*б*)

Поясним причину самого факта наличия оптимального значения τ на одном из рассмотренных выше примеров. Она хорошо видна из различной зависимости от τ двух основных составляющих общего времени решения СЛАУ: времени LU-разложения матрицы $\mathbf{A}^s$, выполняемого только один раз до итераций, и времени на N_{it} последующих итераций (см. рис. 4.3 *б*). Первая составляющая ($T1$, с) пренебрежимо мала слева от оптимального значения τ, но с удалением вправо от него всё более доминирует в общем времени решения СЛАУ. Отметим, что в этой части графика характер зависимости $T1$ от τ полностью совпадает с характером зависимости процента ненулевых элементов (NonZero) матрицы **M**. Вторая составляющая пропорциональна значению N_{it} и, как видно, полностью определяет время решения СЛАУ слева от оптимального значения τ и всё меньше влияет справа. В итоге у середины диапазона τ (где скорость возрастания одной составляющей равна скорости убывания второй) общее время решения СЛАУ минимально.

4.4. Ускорение решения СЛАУ за счет снижения точности вычисления

В данном разделе приводятся результаты исследования работы метода BiCGStab, на примере получения диаграммы направленности антенны, при изменении точности вычислений. Показано, что можно получить ускорение решения СЛАУ за счет снижения точности вычисления до достаточной для получения заданных характеристик [63].

Решение, полученное итерационными методами, в отличие от решения точными методами, получается приближенным. Теоретически при устремлении числа итераций к бесконечности итерационный процесс закончится получением решения, полностью совпадающего с решением, полученным при применении точного метода. Поскольку зачастую нет необходимости в получении точного решения, то дополнительное ускорение итерационного процесса можно достичь за счет снижения точности до достаточной для получения требуемых характеристик. Отметим, что этого в принципе нельзя получить точными методами.

Результаты такого ускорения на примере вычисления диаграммы направленности проводной антенны «чайка» (см. рис. 4.1 *б*) приведены в табл. 4.6. В качестве способа предфильтрации использовался подход (3.55). В первой строке табл. 4.6 приводится время вычисления при решении СЛАУ методом Гаусса (GE), в последующих – итерационным методом с уменьшением заданной точности *Tol*, что соответствует уменьшению числа итераций N_{it}, а в последней строке приводится время, затра-

чиваемое на полное разложение матрицы $\mathbf{A}^s$. Как видно, в этом примере переход от метода Гаусса к итерационному методу с контролем заданной точности значительно ускоряет решение: при $Tol \approx 10^{-7}$ в 10 раз; при $Tol \approx 10^{-2}$ в 15 раз; при $Tol \approx 10^{+2}$ в 20 раз. Между тем, диаграммы направленности для этих случаев практически совпадают (рис. 4.4).

Таблица 4.6

Время решения СЛАУ ($N = 1411$) методом Гаусса и итерационным методом ($\tau = 5 \cdot 10^{-5}$), при уменьшении N_{it} увеличением Tol ($\|\mathbf{r}_k\| < Tol$)

Tol	N_{it}	Время, с
GE	–	267.54
10^{-8}	16	30.65
10^{-7}	14	28.17
10^{-5}	12	25.71
10^{-3}	10	23.23
10^{-2}	8	20.77
10^{0}	6	18.35
10^{+1}	4	15.82
10^{+2}	2	13.34
LU	0	11.42

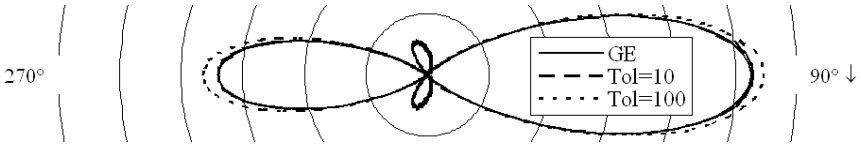


Рис. 4.4. Диаграмма направленности ($|E_\varphi|$, шкала линейная от 0 до 6 В/м) антенны «чайка» в плоскости XY при решении СЛАУ методом Гаусса (GE) и итерационным методом с уменьшением заданной точности Tol ($\|\mathbf{r}_k\| < Tol$)

4.5. Сравнение способов предфильтрации

В данном разделе приведено сравнение способов предфильтрации (3.51)–(3.55). Показано, что выбором способа предфильтрации можно уменьшить время решения СЛАУ до 32%, а в сравнении с методом Гаусса в 12 раз. Выявлена стабильность оптимального значения допуска обнуления в случае использования способа предфильтрации (3.55). Раздел основан на работах [49, 64].

Все вычисления производились в системе TALGAT на примере вычисления токов в трапециевидной зубчатой проводной антенне (рис. 4.1 *а*) на частотах 4, 6, …, 12 ГГц. Использовался стабилизированный метод бисопряженных градиентов (BiCGStab). За начальное приближение вектора решения ($\mathbf{x}_0$) принималось равенство всех его элементов 0.1. Итерации останавливались при условии $\|\mathbf{r}_k\|_2 / \|\mathbf{r}_0\|_2 < 10^{-8}$, где $\mathbf{r}_k$ и $\mathbf{r}_0$ – невязки

после k-й итерации и начального приближения, соответственно, или при достижении 180 итераций. (Для способов (3.51), (3.52) и (3.54) полагалось, что $\varepsilon = \tau$.)

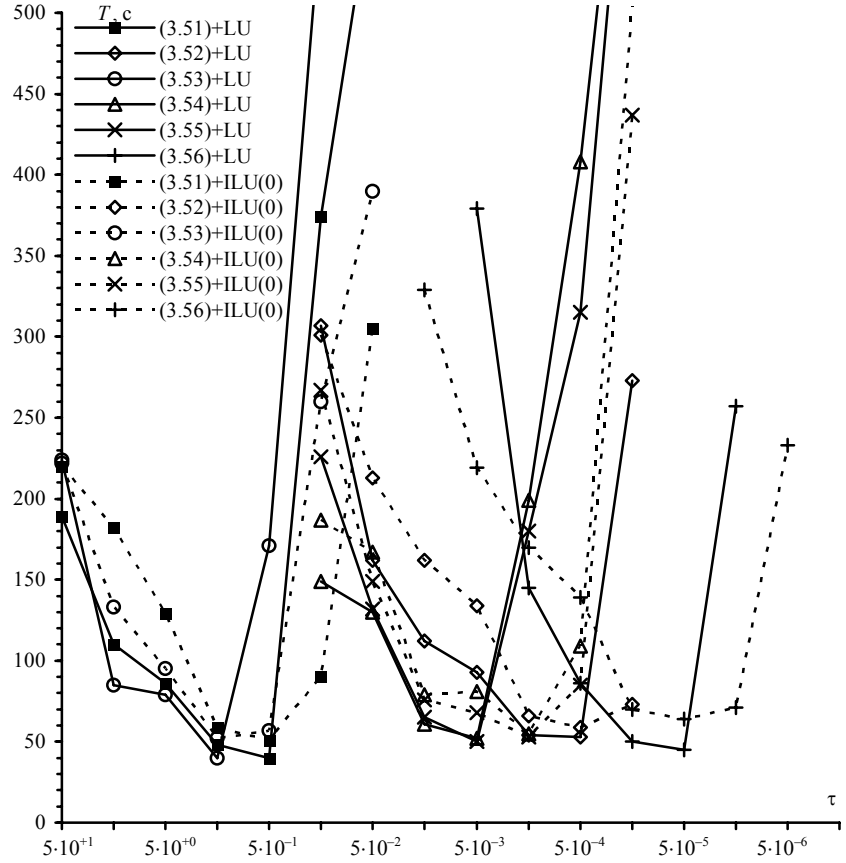


Рис. 4.5. Зависимости времени решения СЛАУ от допуска обнуления при разных способах предфильтрации и предобусловливания для 12 ГГц

В табл. 4.7 для разных частот приведены характеристики решаемой задачи: порядок СЛАУ (N), относительное число ненулевых элементов матрицы СЛАУ (dA), время её заполнения (T_{form}) и время решения методом Гаусса (T_{GE}). Видно, что с увеличением частоты в общих затратах все больше преобладает время решения СЛАУ, что ярко показывает востребованность итерационных методов.

Таблица 4.7

Характеристики матрицы для разных частот

f, ГГц	N	dA, %	T_{form}, с	T_{GE}, с
4	1003	99.9986	7	18
6	1503	99.9989	15	62
8	1981	99.9996	27	142
10	2477	99.9998	42	276
12	2975	99.9997	61	483

Зависимости времени решения СЛАУ итерационным методом от τ при $f = 12$ ГГц для разных способов предфильтрации, выполненных при LU и ILU(0)-разложениях, показаны на рис. 4.5. Из приведенных зависимостей видно, что для оптимальных значений τ использование полного LU-разложения (сплошная линия) предпочтительнее, чем неполного (пунктирная линия), для любого способа предфильтрации. Так, выигрыш варьируется от 5 до 23% для способов (3.54) и (3.53), соответственно.

На рис. 4.6 приведено время решения СЛАУ с предфильтрациями (3.51) – (3.56) (при оптимальных τ) для частот 4, 6, …, 12 ГГц при использовании LU- (*а*) и ILU(0)- (*б*) разложений. Из графиков видно, что на любой частоте (кроме случая использования полного LU-разложения для частоты 10 ГГц) наименьшее время решения СЛАУ дает самый простой способ предфильтрации (3.51), причем вне зависимости от вида разложения для формирования матрицы предобусловливания. Наибольшее расхождение во времени решения СЛАУ при разной предфильтрации наблюдается для частоты 10 ГГц. Так, при использовании полного LU-разложения получен выигрыш до 32% с предфильтрацией (3.52) по отношению к (3.54) и (3.55). В случае же ILU(0)-разложения с предфильтрацией (3.51) выигрыш составляет до 28% относительно тех же способов (3.54) и (3.55) и до 19% по отношению к (3.56).

В табл. 4.8 приведены следующие результаты (при оптимальном значении τ и использовании полного LU-разложения при формировании предобусловливания): время предфильтрации ($T1$), плотность матрицы $\mathbf{A}^s$ (dA^s – число ненулевых элементов матрицы $\mathbf{A}^s$, поделенное на N^2), время формирования матрицы предобусловливания ($T2$), плотность матрицы $\mathbf{M}$ (dM – сумма ненулевых элементов матриц $\mathbf{L}$ и $\mathbf{U}$, поделенная на N^2), общее время решения СЛАУ (T, с) и число итераций (N_{it}), требуемое для достижения заданной точности.

Как видно из табл. 4.8, для любой предфильтрации при оптимальном τ плотность матрицы $\mathbf{A}^s$ составляет менее 1%. На рис. 4.7 для наглядности приведена структура этой матрицы после предфильтрации (3.51) при оптимальном τ. Для других способов заполненность еще меньше.

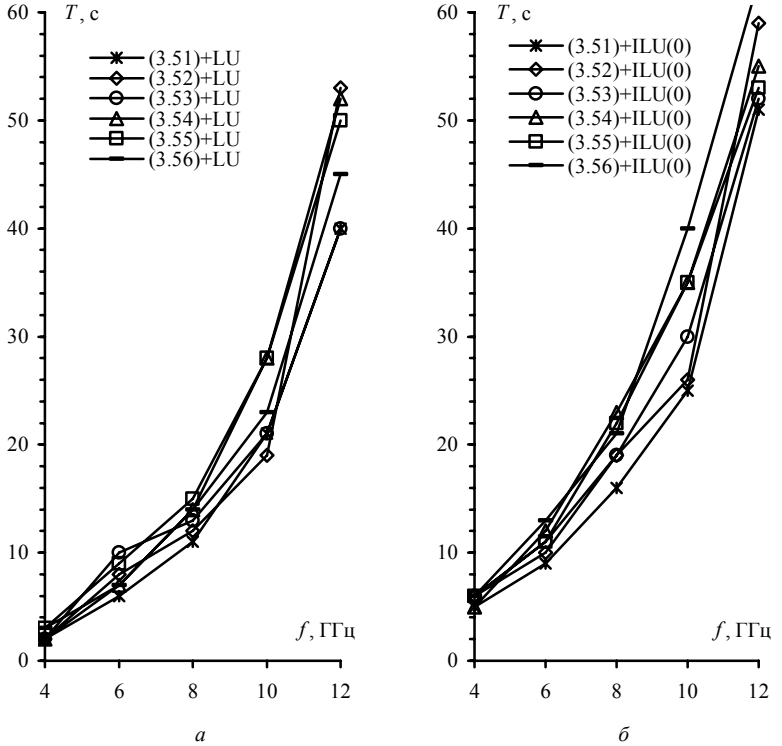


Рис. 4.6. Зависимости минимального времени решения СЛАУ от частоты для различных способов предфильтрации с предобусловливанием, основанным на LU- (*а*) и ILU(0)- (*б*) разложениях

Т а б л и ц а 4.8

Результаты использования различных способов предфильтрации при оптимальных значениях τ для 12 ГГц

Предфильтрация	τ_{opt}	dA^s, %	dM, %	*T*1, с	*T*2, с	*T*, с	N_{it}
(3.51)	$5\cdot10^{-1}$	0.82	2.24	4.15	7.46	40	13
(3.52)	$5\cdot10^{-4}$	0.70	1.59	8.80	14.85	53	15
(3.53)	10^{+0}	0.77	1.73	7.45	11.30	40	13
(3.54)	$5\cdot10^{-3}$	0.59	1.18	7.31	10.48	52	19
(3.55)	$5\cdot10^{-3}$	0.51	0.94	11.25	13.11	50	18
(3.56)	$5\cdot10^{-5}$	0.69	1.55	10.95	13.65	45	15

Из данных табл. 4.7, рис. 4.6 и табл. 4.8 видно увеличение временного выигрыша за счет итерационного метода по сравнению с методом Гаусса

при росте частоты. Так, для 4 ГГц максимальный выигрыш составляет 3 раза, а для 12 ГГц он возрастает до 12 раз. Примечательно, что затраты на решение СЛАУ становятся меньше, чем на формирование матрицы.

В табл. 4.9 приведены оптимальные значения τ по критерию минимизации общего времени решения СЛАУ. Результаты приведены для разных частот для полного LU-разложения и неполного (в скобках) ILU(0)-разложения. Видно, что при изменении частоты для предложенной предфильтрации (3.55) оптимальное значение допуска обнуления остается одним и тем же, чего нельзя сказать о других способах предфильтрации. Способ (3.56) показал стабильность для частот более 4 ГГц. Такая стабильность весьма важна, поскольку она минимизирует затраты времени при многовариантном анализе в большом числе частотных точек или при оптимизации по частоте, широко применяемым на практике.

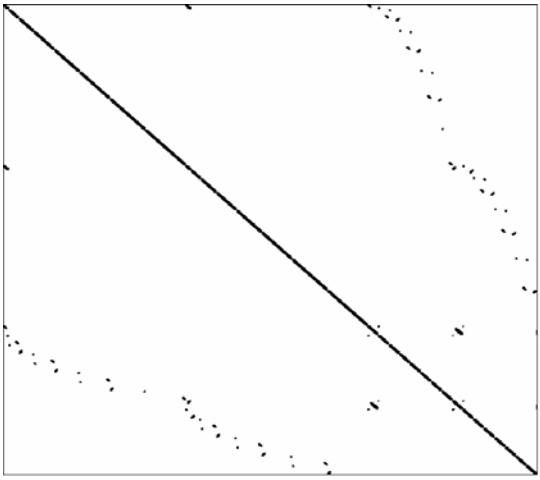


Рис. 4.7. Структура матрицы СЛАУ после предфильтрации (3.51) при $\tau = 5 \cdot 10^{-1}$

Таблица 4.9

Оптимальные значения τ по критерию минимизации времени решения СЛАУ для способов предфильтрации (3.51)–(3.56) с использованием LU и ILU(0) (в скобках) разложений

f, ГГц→	4	6	8	10	12
(3.51)	10^{+0} ($5 \cdot 10^{-1}$)	10^{+0} ($5 \cdot 10^{-1}$)	$5 \cdot 10^{-1}$ ($5 \cdot 10^{-1}$)	$5 \cdot 10^{-1}$ ($5 \cdot 10^{-1}$)	$5 \cdot 10^{-1}$ ($5 \cdot 10^{-1}$)
(3.52)	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ (10^{-3})	10^{-3} ($5 \cdot 10^{-4}$)	$5 \cdot 10^{-4}$ ($5 \cdot 10^{-4}$)	$5 \cdot 10^{-4}$ ($5 \cdot 10^{-4}$)
(3.53)	10^{+0} (10^{+0})	$5 \cdot 10^{+0}$ (10^{+0})	10^{+0} (10^{+0})	10^{+0} (10^{+0})	10^{+0} (10^{+0})
(3.54)	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ ($5 \cdot 10^{-3}$)	$5 \cdot 10^{-3}$ ($5 \cdot 10^{-3}$)	$5 \cdot 10^{-3}$ ($5 \cdot 10^{-3}$)	$5 \cdot 10^{-3}$ (10^{-3})
(3.55)	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ (10^{-3})	$5 \cdot 10^{-3}$ (10^{-3})
(3.56)	$5 \cdot 10^{-4}$ ($5 \cdot 10^{-5}$)	$5 \cdot 10^{-5}$ ($5 \cdot 10^{-5}$)	$5 \cdot 10^{-5}$ ($5 \cdot 10^{-5}$)	$5 \cdot 10^{-5}$ ($5 \cdot 10^{-4}$)	$5 \cdot 10^{-5}$ ($5 \cdot 10^{-5}$)

При сведении интегральных уравнений к СЛАУ вычислительные затраты идут на формирование матрицы и собственно на решение линейной системы. Затраты на формирование можно снизить за счет использования быстрых аналитических формул вместо точного, но вычислительно затратного численного интегрирования. Другим ресурсом снижения вычислительных затрат является корректный выбор детальности дискретизации для конкретной задачи. Наименьшей приемлемой дискретизацией полагают λ/10 [65]. Увеличение дискретизации даёт более точные результаты, но каждое удвоение детальности дискретизации удваивает порядок N матрицы СЛАУ. Это увеличивает затраты памяти на её хранение и общее время её формирования в 4 раза (поскольку она состоит из N^2 элементов) и увеличивает время решения СЛАУ методом Гаусса в 8 раз (поскольку оно $\sim N^3$). Очевидно, что корректный выбор детальности дискретизации существенно сокращает вычисления. Наконец, если ресурсы снижения порядка матрицы и уменьшения времени её заполнения исчерпаны, и основные вычислительные затраты связаны с решением СЛАУ, то можно ускорить её решение с помощью итерационных методов.

В табл. 4.10 для разной дискретизации приведены характеристики решаемой задачи: порядок СЛАУ (N), относительное число ненулевых элементов матрицы СЛАУ (dA), время её заполнения (T_{form}) и время решения методом Гаусса (T_{GE}). Видно, что с увеличением дискретизации в общих затратах все больше преобладает время решения СЛАУ. Таким образом, очевидна востребованность итерационных методов.

Таблица 4.10

Характеристики матрицы при разной дискретизации для частоты 3 ГГц

Дискретизация	N	dA, %	T_{form}, с	T_{GE}, с
λ/10	765	99.9974	4	8
λ/20	1513	99.9991	16	63
λ/30	2247	99.9997	35	206
λ/40	2997	99.9997	62	497

На рис. 4.8 приведено время решения СЛАУ с предфильтрациями (3.51)–(3.56) (при оптимальных τ) в зависимости от дискретизации для частоты 3 ГГц. Из графиков трудно выделить явное преимущество какого-либо способа предфильтрации. Однако расхождение во времени решения СЛАУ при разных способах предфильтрации оказывается существенным. Так выигрыш способа (3.51) по отношению к (3.54) при λ/40 достигает 30%. Таким образом, совершенствование предфильтрации актуально даже только из соображений уменьшения времени решения СЛАУ, а также, как будет показано далее, и по другим критериям.

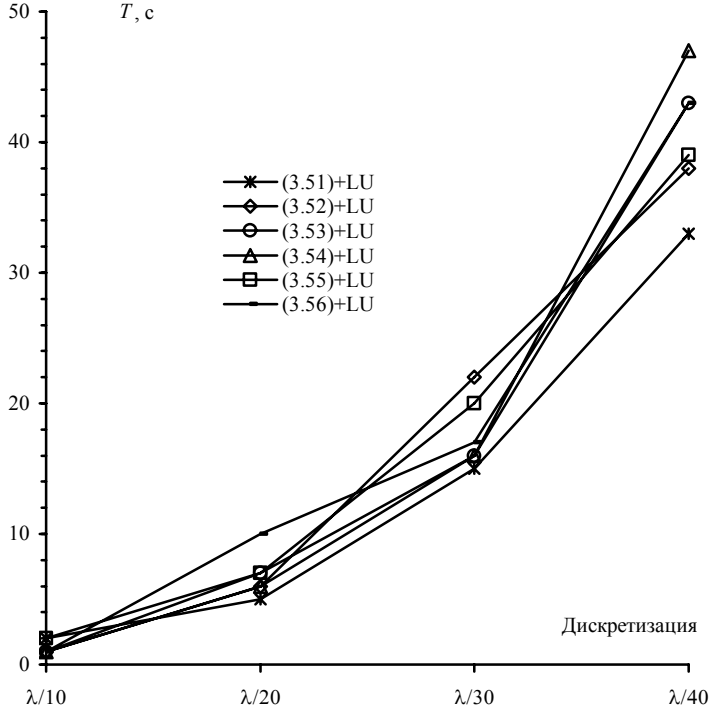


Рис. 4.8. Зависимости минимального времени решения СЛАУ от дискретизации для различных способов предфильтрации

Из данных табл. 4.10 и рис. 4.8 можно оценить увеличение временного выигрыша за счет итерационного метода по сравнению с методом Гаусса при увеличении дискретизации: он возрастает от 8 при λ/10 до 15 при λ/40. Примечательно, что затраты на решение СЛАУ становятся меньше, чем на формирование матрицы.

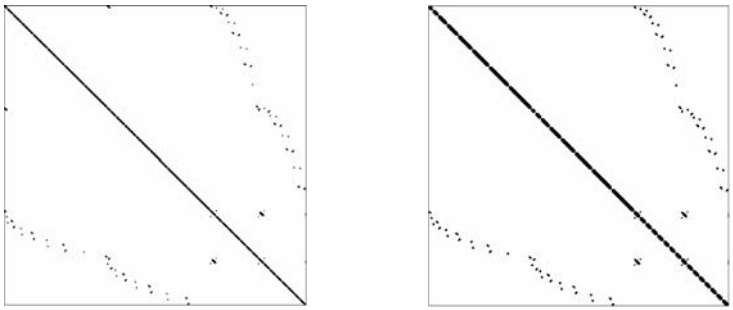
В табл. 4.11 приведены результаты использования различных способов предфильтрации (при оптимальном значении τ): время предфильтрации ($T1$), плотность матрицы $\mathbf{A}^s$ (dA^s – число ненулевых элементов матрицы $\mathbf{A}^s$, поделенное на N^2), полное время (включая $T1$) формирования матрицы предобусловливания ($T2$), плотность матрицы **M** (dM – сумма ненулевых элементов матриц **L** и **U**, поделенная на N^2), общее время решения СЛАУ (T, с) и число итераций (N_{it}), требуемое для достижения заданной точности.

Таблица 4.11

Результаты использования различных способов предфильтрации при оптимальных значениях τ для λ/40 (3 ГГц)

Предфильтрация	τ_{opt}	dA^s, %	dM, %	$T1$, с	$T2$, с	T, с	N_{it}
(3.51)	10^{-1}	1.69	3.34	3.82	9.65	33	8
(3.52)	10^{-4}	1.78	3.53	7.84	13.80	38	8
(3.53)	$5{\cdot}10^{-1}$	0.95	1.45	8.70	11.47	43	14
(3.54)	$5{\cdot}10^{-4}$	0.82	1.20	7.51	9.94	47	15
(3.55)	10^{-4}	1.68	3.30	11.50	17.17	39	8
(3.56)	$5{\cdot}10^{-6}$	1.01	1.58	10.97	13.86	43	13

Как было показано в предыдущем разделе (см. табл. 4.8), для любой предфильтрации при оптимальном τ плотность матрицы $\mathbf{A}^s$ составляет менее 1%. В случае достижения примерно того же N, но за счет дискретизации (табл. 4.11) матрица является более плотной для любой предфильтрации. На рис. 4.9 для наглядности сравнены структуры этой матрицы после предфильтрации (3.51) при оптимальном τ для случая λ/10 (12 ГГц) – (*а*) и λ/40 (3 ГГц) – (*б*). Видно, что увеличение дискретизации при оптимальном допуске обнуления приводит к более выраженным ненулевым областям в матрице $\mathbf{A}^s$.



а *б*

Рис. 4.9. Структура матрицы СЛАУ после предфильтрации (3.51): для λ/10 (12 ГГц) при $\tau = 5{\cdot}10^{-1}$ (*а*) и λ/40 (3 ГГц) при $\tau = 10^{-1}$ (*б*)

В табл. 4.12 приведены оптимальные значения τ по критерию минимизации общего времени решения СЛАУ при разной дискретизации для 3 ГГц. Видно, что ни один из способов не показал высокой стабильности оптимального допуска обнуления при изменении дискретизации во всем диапазоне. Способы (3.53) и (3.54) показали стабильность при дискретизации более чем λ/10. Для способов (3.52), (3.55) шаг изменения дискретизации соответствует примерно шагу изменения оптимального допуска обнуления.

Таблица 4.12

Оптимальные значения τ по критерию минимизации времени решения СЛАУ для способов предфильтрации (3.51)–(3.56) при изменении дискретизации

Предфильтрация	λ/10	λ/20	λ/30	λ/40
(3.51)	$5\cdot10^{+0}$	$5\cdot10^{-1}$	$5\cdot10^{-1}$	10^{-1}
(3.52)	$5\cdot10^{-3}$	10^{-3}	$5\cdot10^{-4}$	10^{-4}
(3.53)	10^{+0}	$5\cdot10^{-1}$	$5\cdot10^{-1}$	$5\cdot10^{-1}$
(3.54)	$5\cdot10^{-3}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$
(3.55)	$5\cdot10^{-3}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	10^{-4}
(3.56)	10^{-4}	$5\cdot10^{-5}$	$5\cdot10^{-6}$	$5\cdot10^{-6}$

Из табл. 4.8 и 4.11 видно, что в большинстве случаев в общем времени формирования матрицы предобусловливания ($T2$) преобладает часть, связанная с предфильтрацией ($T1$). Таким образом, одним из способов снижения времени решения СЛАУ является снижение временных затрат на предфильтрацию. Так, решение для случая λ/40 (3 ГГц) за счет поиска только на главной диагонали было сокращено с 38 с (при использовании способа (3.52)) до 33 с (при использовании усовершенствованного (3.52)) при неизменных значениях dA^{s}, dM и N_{it}. Таким образом, удалось сократить время решения СЛАУ на 13%. Ту же задачу (λ/40, 3 ГГц) при использовании подхода (3.57) удалось решить за 8 итераций и 35 с. В табл. 4.13 приведены результаты более детального исследования этого подхода. Видно, что шагу изменения дискретизации соответствует шаг изменения оптимального допуска обнуления.

Таблица 4.13

Оптимальные значения τ по критерию минимизации времени решения СЛАУ для предфильтрации (3.57)

Дискретизация	λ/10	λ/20	λ/30	λ/40
τ_{opt}	$5\cdot10^{-3}$	10^{-3}	$5\cdot10^{-4}$	10^{-4}

Далее приведем результаты подобных вычислений для другой, широкодиапазонной проводной, антенны (см. рис. 4.1 *в*) при изменении частоты и дискретизации. Как и ранее, использовался метод BiCGStab, за начальное приближение вектора решения принималось равенство всех его элементов 0.1, итерации останавливались при условии $\|\mathbf{r}_k\|_2/\|\mathbf{r}_0\|_2 < 10^{-8}$.

В табл. 4.14 для разных частот приведены характеристики решаемой задачи: порядок СЛАУ (N), относительное число ненулевых элементов матрицы СЛАУ (dA), время её заполнения (T_{form}) и время решения методом Гаусса (T_{GE}). Аналогичные результаты для разной дискретизации приведены в табл. 4.15. Видно, что с увеличением как частоты, так и дискретизации в общих затратах все больше преобладает время решения СЛАУ. Опять же, очевидна востребованность итерационных методов.

Таблица 4.14

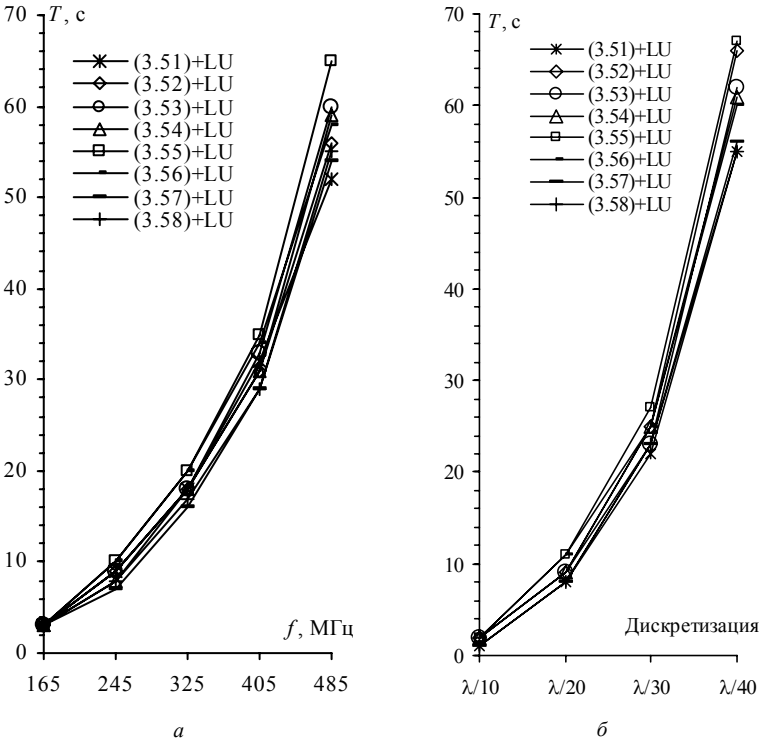
Характеристики матрицы для разных частот

f, МГц	N	dA, %	T_{form}, с	T_{GE}, с
165	991	100	7	17
245	1473	100	15	58
325	1951	100	26	136
405	2431	100	41	262
485	2909	100	59	464

Таблица 4.15

Характеристики матрицы при разной дискретизации для частоты 125 МГц

Дискретизация	N	dA, %	T_{form}, с	T_{GE}, с
λ/10	753	100	4	7
λ/20	1501	100	15	62
λ/30	2251	100	35	209
λ/40	2999	100	63	500



а *б*

Рис. 4.10. Зависимости минимального времени решения СЛАУ для различных способов предфильтрации от частоты (*а*) и от дискретизации (*б*)

Из данных табл. 4.14, 4.15 и рис. 4.10 можно оценить увеличение временного выигрыша за счет итерационного метода по сравнению с методом Гаусса при росте частоты и дискретизации. Так, для 165 МГц максимальный выигрыш составляет 5 раз, а для 485 МГц он возрастает до 9 раз. За счет же дискретизации выигрыш во времени возрастает от 7 при $\lambda/10$ до 9 при $\lambda/40$. Примечательно, что, как и для трапециевидной зубчатой антенны, затраты на решение СЛАУ становятся меньше, чем на формирование матрицы.

Таблица 4.16

Результаты использования различных способов предфильтрации при оптимальных значениях τ для 485 МГц при $\lambda/40$

Предфильтрация	τ_{opt}	dA^s, %	dM, %	$T1$, с	$T2$, с	T, с	N_{it}
(3.51)	10^{-1}	2.51	8.31	3.69	25.89	52	12
(3.52)	$5\cdot10^{-4}$	1.46	5.62	7.49	21.97	56	20
У (3.52)	$5\cdot10^{-4}$	1.47	5.62	3.69	18.82	53	20
(3.53)	$5\cdot10^{-1}$	1.50	5.78	7.54	23.28	60	18
(3.54)	$5\cdot10^{-4}$	1.47	5.63	7.33	22.64	59	20
(3.55)	$5\cdot10^{-4}$	1.19	5.07	11.12	25.18	65	22
(3.56)	$5\cdot10^{-4}$	1.81	6.47	10.79	27.71	58	16
(3.57)	$5\cdot10^{-4}$	1.47	5.62	3.96	19.29	54	20
(3.58)	$5\cdot10^{-4}$	1.47	5.62	4.19	19.98	55	20

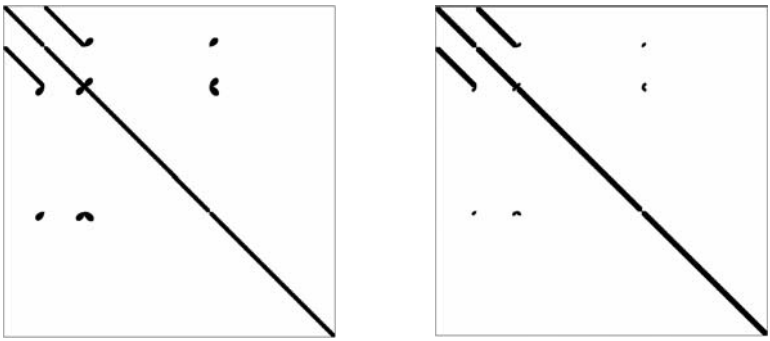
В табл. 4.16 для 485 МГц при $\lambda/10$ приведены следующие результаты (при оптимальном значении τ и использовании полного LU-разложения при формировании предобусловливания): время предфильтрации ($T1$), плотность матрицы $\mathbf{A}^s$ (dA^s – число ненулевых элементов матрицы $\mathbf{A}^s$, поделенное на N^2), время формирования матрицы предобусловливания ($T2$), плотность матрицы $\mathbf{M}$ (dM – сумма ненулевых элементов матриц $\mathbf{L}$ и $\mathbf{U}$, поделенная на N^2), общее время решения СЛАУ (T, с) и число итераций (N_{it}), требуемое для достижения заданной точности. В четвертой строке приведены результаты для усовершенствованного способа (3.52) (У (3.52)). Аналогичные результаты для частоты 125 МГц при дискретизации $\lambda/40$ приведены в табл. 4.17.

Как видно из табл. 4.16, для любой предфильтрации при оптимальном τ плотность матрицы $\mathbf{A}^s$ составляет менее 3%. В случае достижения примерно того же N, но за счет дискретизации (табл. 4.17) матрица является более плотной для любой предфильтрации. На рис. 4.11 для наглядности приведена структура этой матрицы после предфильтрации (3.51) при оптимальном τ для случая $\lambda/10$ (485 МГц) – (*а*) и $\lambda/40$ (125 МГц) – (*б*). Видно, что увеличение дискретизации при оптимальном допуске обнуления приводит к более выраженным ненулевым областям в матрице $\mathbf{A}^s$ вблизи главной диагонали.

Таблица 4.17

Результаты использования различных способов предфильтрации при оптимальных значениях τ для λ/40 при 125 МГц

Предфильтрация	τ_{opt}	dA^s, %	dM, %	$T1$, с	$T2$, с	T, с	N_{it}
(3.51)	$5\cdot10^{-2}$	2.89	6.93	4.91	24.57	55	12
(3.52)	10^{-4}	1.93	5.34	8.04	23.21	57	16
У (3.52)	10^{-4}	1.93	5.34	4.02	19.52	53	16
(3.53)	$5\cdot10^{-2}$	2.35	6.01	8.22	25.30	62	14
(3.54)	$5\cdot10^{-5}$	1.93	5.34	7.75	23.26	61	16
(3.55)	10^{-5}	1.76	5.12	11.72	26.52	67	17
(3.56)	$5\cdot10^{-7}$	2.43	6.11	11.22	28.19	60	13
(3.57)	$5\cdot10^{-5}$	1.93	5.34	5.90	21.32	56	16
(3.58)	$5\cdot10^{-5}$	1.93	5.34	5.24	20.98	55	16



а *б*

Рис. 4.11. Структура матрицы СЛАУ после предфильтрации (3.51): для λ/10 (485 МГц) при $\tau = 10^{-1}$ (*а*) и λ/40 (125 МГц) при $\tau = 5\cdot10^{-2}$ (*б*)

Таблица 4.18

Оптимальные значения τ по критерию минимизации времени решения СЛАУ для способов предфильтрации (3.51)–(3.58) при изменении частоты

$f \rightarrow$	165	245	325	405	485
(3.51)	$5\cdot10^{-1}$	10^{-1}	10^{-1}	10^{-1}	10^{-1}
(3.52)	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$
(3.53)	$5\cdot10^{-1}$	$5\cdot10^{-1}$	$5\cdot10^{-1}$	$5\cdot10^{-1}$	$5\cdot10^{-1}$
(3.54)	$5\cdot10^{-4}$	10^{-3}	$5\cdot10^{-4}$	10^{-3}	$5\cdot10^{-4}$
(3.55)	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$
(3.56)	10^{-4}	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$
(3.57)	10^{-3}	10^{-3}	$5\cdot10^{-4}$	$5\cdot10^{-4}$	$5\cdot10^{-4}$
(3.58)	10^{-3}	10^{-3}	10^{-3}	10^{-3}	$5\cdot10^{-4}$

В табл. 4.18 приведены оптимальные значения τ по критерию минимизации общего времени решения СЛАУ для разных частот. Аналогич-

ные результаты при разной дискретизации для 125 МГц приведены в табл. 4.19. Видно, что при изменении частоты для способов предфильтрации (3.52), (3.53) и (3.55) оптимальное значение допуска обнуления остается одним и тем же, чего нельзя сказать о других способах предфильтрации. Способы (3.51) и (3.56) показали стабильность для частот более 165 МГц. При изменении дискретизации только способ (3.52) показал стабильность оптимального допуска обнуления. Для способа (3.55), как и ранее для трапециевидной зубчатой антенны, шаг изменения дискретизации соответствует шагу изменения оптимального допуска обнуления.

Таблица 4.19

Оптимальные значения τ по критерию минимизации времени решения СЛАУ для способов предфильтрации (3.51)–(3.58) при изменении дискретизации

Предфильтрация	λ/10	λ/20	λ/30	λ/40
(3.51)	$5\cdot10^{-1}$	10^{-1}	10^{-1}	$5\cdot10^{-2}$
(3.52)	10^{-4}	10^{-4}	10^{-4}	10^{-4}
(3.53)	$5\cdot10^{-2}$	$5\cdot10^{-2}$	10^{-1}	$5\cdot10^{-2}$
(3.54)	$5\cdot10^{-4}$	10^{-4}	10^{-4}	$5\cdot10^{-5}$
(3.55)	$5\cdot10^{-4}$	10^{-4}	$5\cdot10^{-5}$	10^{-5}
(3.56)	$5\cdot10^{-6}$	10^{-6}	10^{-6}	$5\cdot10^{-7}$
(3.57)	$5\cdot10^{-4}$	10^{-4}	10^{-4}	$5\cdot10^{-5}$
(3.58)	$5\cdot10^{-4}$	10^{-4}	10^{-4}	$5\cdot10^{-5}$

Таким образом, итерационные методы позволяют значительно уменьшить время решения СЛАУ с плотной матрицей по сравнению с методом Гаусса. Как было показано, затраты на решение СЛАУ становятся меньше, чем на формирование матрицы.

ЗАКЛЮЧЕНИЕ

Монография посвящена проблеме решения системы линейных алгебраических уравнений. Авторы надеются, что их работа окажется полезной для читателя. Общий обзор проблемы с многочисленными ссылками на работы известных исследователей, может стать хорошим стартом для новых исследований этой задачи.

Приведено детальное рассмотрение большого количества прямых и итерационных методов, как классических, так и наиболее работоспособных на данный момент. Рассмотрены предфильтрация и предобусловливание, позволяющие добиться ускорения итерационного решения. Программная реализация этих методов выполнена в виде модуля матричных операций MATRIX единой системы компьютерного моделирования и уже используется. Так, реализованный модуль использовался для оценки паразитных электромагнитных эффектов в печатных платах и в кабелях аппаратуры, разрабатываемой в НПЦ «Полюс». Результаты работы внедрены в систему компьютерного моделирования электромагнитной совместимости при выполнении инновационной программы ТУСУРа и в учебный процесс ТУСУРа. Также реализованные итерационные методы использованы при выполнении проекта 06-08-01242, поддержанного Российским фондом фундаментальных исследований.

Исследованные возможности уменьшения времени решения СЛАУ итерационными методами за счёт соответствующего выбора способа предфильтрации, ее параметров, способа предобусловливания на примере определения токов в проводной антенне показали, что можно добиться ускорения решения СЛАУ по сравнению с методом Гаусса в десятки раз. В работе эти возможности показаны на нескольких конфигурациях антенн, но они не исчерпаны полностью. Так, выполненное сравнение способов алгебраической предфильтрации при решении СЛАУ итерационными методами позволяет оценить изменение оптимального значения порога/допуска обнуления для каждого из способов, на примере определения токов в проводной антенне, как при изменении частоты, так и при изменении дискретизации.

ЛИТЕРАТУРА

1. *Амосов А.А.*, *Дубинская Ю.А.*, *Копченова Н.В.* Вычислительные методы для инженеров. – М.: МЭИ, 2003.
2. *Пирумов У.Г.* Численные методы. – М.: Дрофа, 2003.
3. *В.М Вержбицкий.* Основы численных методов. – М.: Высшая школа, 2002.
4. *Форсайт Дж.*, *Молер К.* Численное решение систем линейных алгебраических уравнений. – М.: Мир, 1969.
5. *Райс Дж.* Матричные вычисления и математическое обеспечение. – М.: Мир, 1984.
6. *Райс Дж.*, *Малькольм М.*, *Моулер К.* Машинные методы математических вычислений. – М.: Мир, 1980.
7. *Канахер Д.*, *Моулер К.*, *Нэш С.* Численные методы и программное обеспечение. – М.: Мир, 1999.
8. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. – М.: Мир, 1991.
9. *Ильин В.П.*, *Кузнецов Ю.И.* Трехдиагональные матрицы и их приложения. – М.: Наука, 1985.
10. *Самарский А.А.*, *Николаев Е.С.* Методы решения сеточных уравнений. – М.: Наука, 1978.
11. *Бахвалов Н.С.*, *Жибков Н.П.*, *Кобельков Г.М.* Численные методы. – М.: Наука, 1987.
12. *Голуб Дж.*, *Ван Лоун Ч.* Матричные вычисления. – М.: Мир, 1999.
13. *Икрамов Х.Д.* Вычислительные методы линейной алгебры. (Решение больших разреженных систем уравнений прямыми методами). – М.: Знание, 1989.
14. *Джордж А.*, *Лю Дж.* Численное решение больших разреженных систем уравнений. – М.: Мир, 1984.
15. *Писсанецки С.* Технология разреженных матриц. – М.: Мир, 1988.
16. *Эстербю О.*, *Златаев З.* Прямые методы для разреженных матриц. – М.: Мир, 1987.
17. *Saad Y.*, *Henk A.*, *van der Vorst H.* Iterative solution of linear systems in the 20-th century // Journal of Computational and Applied Mathematics. 2000. Vol. 123, № 1. P. 1–33.
18. *Ильин В.П.* Линейная алгебра: от Гаусса до суперкомпьютеров будущего // Природа. 1999. № 6. С. 11–18.
19. *Колотилина Л.Ю.* Явно предобусловленные системы линейных алгебраических уравнений с плотной матрицей // Советская математика. 1988. № 43. С. 2566–2573.
20. *Баландин М.Ю., Шурина Э.П.* Методы решения СЛАУ большой размерности. – Новосибирск: НГТУ, 2000.
21. *Barrett R.*, *Berry M.*, *Chan T.F.*, *Demmel J.*, *Donato J.*, *Dongarra J.*, *Eijkhout V.*, *Pozo R.*, *Romine C.*, *van der Vorst H.* Templates for the solution of linear systems: building block for iterative methods. SIAM. Philadelphia. 1994.
22. *Saad Y.* Iterative methods for sparse linear systems. Second edition. SIAM, Philadelphia. 2000.
23. *Van der Vorst H.* Iterative Krylov methods for large linear systems. Cambridge Monographs on Applied and Computational Mathematics, 2003.
24. *Axellson O.* Iterative Solution Methods. Cambridge University Press. New York, 1994.
25. *Ильин В.П.* Методы неполной факторизации для решения линейных систем. – М.: Физмат, 1995.
26. *Saad Y.*, ILUT: a dual threshold incomplete LU factorization // Numer. Linear Algebra Appl. 1994. Vol. 1, № 4. P. 387–402.
27. *Saad Y.*, *Zhang J.* BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices // SIAM J. Matrix Anal. Appl. 1999. Vol. 21, № 1. P. 279–299.

28. *Li N.*, *Saad Y.*, *Chow E.* Crout Versions of ILU for General Sparse Matrices // SIAM Journal on Scientific Computing. 2003. Vol. 25, № 2. P. 716–728.
29. *Lee J.*, *Zhang J.*, *Lu C.* Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems // J. of Comput. Phys. 2003. № 185. P. 158–175.
30. *Bangtsson E.*, *Neytcheva M.* Approaches to reduce the computational cost when solving linear systems of equations arising in Boundary Element Method discretizations. Uppsala University, Department of Information Technology, TR/2003–053.
31. *Antonini G.*, *Orlandi A.*, *Ruehli A.* Fast iterative solution for the wavelet-PEEC method // Proc. of 14th International Zurich Symposium on EMC, Zurich, Switzerland. 2001. P. 545–550.
32. *Rahola J.*, *Tissari S.* Iterative solution of dense linear systems arising from the electrostatic integral equation in MEG // Physics in medicine and biology. 2002. № 47. P. 961–975.
33. *Zunoubi M.R.* A combined BI-CGSTAB(*l*) and wavelet transform method for EM problems using method of moments // Progress in electromagnetics research. 2005. № 52. P. 205–224.
34. *Carpentieri B.*, *Duff I.S.*, *Giraud L.* Some sparse pattern selection strategies for robust Flobenius norm minimization preconditioners in electromagnetism. RAL-TR-2000-009.
35. *Benzi M.*, *Tuma M.* Numerical experiments with two approximate inverse preconditioners. CERFACS TR/PA/97/11.
36. *Benzi M.*, *Tuma M.* A comparative study of sparse approximate inverse preconditioners. Los Alamos national laboratory technical report LA-UR-98-0024.
37. *Chow E.*, *Saad Y.* Approximate inverse preconditioners via sparse-sparse iterations // SIAM J. Sci. Comput. 1998. Vol. 19. P. 995–1023.
38. *Grote M.*, *Huckle T.* Parallel preconditioning with sparse approximate inverses // SIAM J. Sci. Comput. 1997. № 18. P. 838–853.
39. *Kolotilina L.Yu.*, *Yeremin A.Yu.* Factorized sparse approximate inverse preconditioning I. Theory // SIAM J. Matrix Anal. Appl. 1993. № 14. P. 45–58.
40. *Ewe W.-B.*, *Li L.-W.*, *Wu Q.*, *Leong M.-S.* Preconditioners for adaptive integral method implementation // IEEE Transactions on Antennas and Propagation. 2005. Vol. 53, № 7. P. 2346–2350.
41. *Alleon G.*, *Benzi M.*, *Giraud L.* Sparse Approximate Inverse Preconditioning for dense Linear Systems Arising in Computation of Electromagnetics. CERFACS TR/PA/97/05.
42. *Carpentieri B.*, *Duff I.S.*, *Giraud L.*, *Made M.M.* Sparse symmetric preconditioners for dense linear systems in electromagnetism. CERFACS TR/PA/01/35.
43. *Wagner R.L.*, *Chew W.C.* A stydy of wavelets for the solution of electromagnetic integral equations // IEEE Transactions on Antennas and Propagation. 1995. Vol. 43, № 8. P. 802–810.
44. *Chen K.* Discrete wavelet transforms accelerated sparse preconditioners for dense boundary element systems // Electronic transactions on numerical analysis. 1999. Vol. 8. P. 138–153.
45. *Ford J.M.* An improved DWT-based preconditioner for dense matrix problem. Manchester centre for computational mathematics numerical analysis reports. 2002. Numerical analysis report № 412.
46. *Prakash V.V.S.*, *Mittra R.* An efficient preconditioner for iterative solvers // Turk J. Elec. Engin. 2002. Vol. 10, № 2. P. 371–375.
47. *Sertel K.*, *Volakis J.L.* Incomplete LU preconditioner for FMM implementations. 2000 Applied Computational Electromagnetics Society, Monterey, CA. P. 859–866.
48. *Газизов Т.Р.*, *Куксенко С.П.* Оптимизация допуска обнуления при решении СЛАУ итерационными методами с предобусловливанием в задачах вычислительной электродинамики // Электромагнитные волны и электронные системы. 2004. № 8. С. 26–28.

49. *Куксенко С.П.*, *Газизов Т.Р.* Совершенствование способов предфильтрации для решения СЛАУ с плотной матрицей итерационными методами с предобусловливанием в задачах вычислительной электродинамики // Электромагнитные волны и электронные системы. 2007. № 9. С. 12–17.
50. *Saad Y., Schultz M.H.* GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems // SIAM J. Sci. and Stat. Comput. 1986. № 7. P. 856–869.
51. *Fletcher R.* Conjugate gradient methods for indefinite systems // Watson G.A., editor, Proccedings of Dundee Biennal Conference on Numerical Analysis. 1974. P. 73–89.
52. *Freund R.W.* A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems // SIAM J. Sci. Comput. 1993. № 14. P. 470–482.
53. *Vorst van der H.* Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of nonsymmetric linear systems // SIAM J. Sci. and Stat. Comput. 1992. № 13. P. 631–644.
54. *Freund R.W., Nachtigal N.M.* QMR: a quasi-minimal residual method for non-hermitian linear system // Num. Math. 1991. № 60. P. 315–339.
55. *Sonneveld P.* CGS, a fast Lanczos-type solver for nonsymmetric linear systems // SIAM J. Sci. Statist. Comput. 1989. № 10. P. 36–52.
56. *Hestenes M.R., Stiefel E.* Methods of gradients for solving linear systems // Res. Natl. Bur. Stand. 1952. № 49. P. 409–436.
57. *Фадеев Д.К.*, *Фадеева В.Н.* Вычислительные методы линейной алгебры. – М.: ФМ, 1963.
58. *Golub G.H.*, *Van Loan C.F.* Matrix computation. Third edition. The Johns Hopkins University Press, 1996.
59. *Газизов Т.Р.* и др. Комплексная оптимизация генетическими алгоритмами для обеспечения ЭМС. Сб. науч. докл. VI Межд. симп. по электромагнитной совместимости и электромагнитной экологии. СПб., 2005. С. 160–164.
60. *Kominami M.*, *Rokushima K.* On the integral equation of piecewise linear antennas // IEEE Trans. Antennas Propagat. 1981. Vol. 29. P. 787–792.
61. *Рыбин А.П.*, *Лощилов А.Г.*, *Малютин Н.Д.* Моделирование и экспериментальное исследование широкополосных антенн в ДКМВ диапазоне. Сборник научных трудов всеросс. науч.-техн. конф. «Научная сессия ТУСУР – 2004». 2004. С. 122–125.
62. *Куксенко С.П.*, *Газизов Т.Р.* Оптимизация параметров стабилизированного метода бисопряжённых градиентов при решении задач вычислительной электродинамики. Материалы Шестой Всеросс. науч.-практ. конф. «Проблемы информационной безопасности государства, общества и личности». Томск, 2004. С. 113–115.
63. *Куксенко С.П.*, *Газизов Т.Р.* Ускорение решения СЛАУ в задачах вычислительной электродинамики. Материалы Седьмой Всеросс. науч.-практ. конф. «Проблемы информационной безопасности государства, общества и личности». Томск, 2005. С. 54–57.
64. *Газизов Т.Р.*, *Куксенко С.П.* Сравнение способов предфильтрации при решении СЛАУ с плотной матрицей итерационными методами с предобусловливанием // Инфокоммуникационные технологии. 2007. Том 5, № 2. С. 14–18.
65. *Харрингтон Р.Ф.* Применение матричных методов к задачам теории поля // Труды ИИЭР. 1967. № 2. С. 5–19.