ISSN 2686-9667. Вестник российских университетов. Математика

Том 24, № 128 2019

DOI 10.20310/2686-9667-2019-24-128-393-431
УДК 519.6, 519.85

Универсальные алгоритмы решения дискретных стационарных уравнений Беллмана

Григорий Лазаревич ЛИТВИНОВ[1], **Анатолий Яковлевич РОДИОНОВ**[2], **Сергей СЕРГЕЕВ**[3], **Андрей Николаевич СОБОЛЕВСКИЙ**[1]

[1] ФГБУН «Институт проблем передачи информации им. А. А. Харкевича Российской академии наук»
127051, Российская Федерация, г. Москва, Большой Каретный переулок, 19
[2] Московский центр непрерывного математического образования
119002, Российская Федерация, г. Москва, Большой Власьевский переулок, 11
[3] Университет Бирмингема, Школа Математики
B15 2TT Великобритания, г. Бирмингем, Здание Ватсона

Universal algorithms for solving discrete stationary Bellman equations

Grigory L. LITVINOV[1], **Anatoliy Ya. RODIONOV**[2], **Serge SERGEEV**[3], **Andrei N. SOBOLEVSKY**[1]

[1] Institute for Information Transmission Problems of the Russian Academy of Sciences
19 Bolshoy Karetny Pereulok, Moscow 127051, Russian Federation
[2] Moscow Center for Continuous Mathematical Education
11 Bolshoy Vasil'evsky Pereulok, Moscow 119002, Russian Federation
[3] University of Birmingham, School of Mathematics
Watson Building, Birmingham B15 2TT, United Kingdom

Аннотация. В настоящей работе исследуются алгоритмы решения дискретных стационарных (или) матричных уравнений Беллмана над полукольцами, в особенности над тропическими и идемпотентными полукольцами. Также приведены оригинальные алгоритмы, приложения и программная реализация.

Ключевые слова: тропическая линейная алгебра; идемпотентные полукольца; матричные уравнения Беллмана; универсальные алгоритмы

Благодарности: Работа выполнена при поддержке РФФИ (проекты № 11-01-93106_а, № 12-01-00886_а), грантом EPSRC RRAH15735.

Для цитирования: *Литвинов Г.Л., Родионов А.Я., Сергеев С.Н., Соболевский А.Н.* Универсальные алгоритмы решения дискретных стационарных уравнений Беллмана // Вестник российских университетов. Математика. 2019. Т. 24. № 128. С. 393–431. DOI 10.20310/2686-9667-2019-24-128-393-431.

Abstract. This paper investigates algorithms for solving discrete stationary (or) matrix Bellman equations over semirings, in particular over tropical and idempotent semirings, Also there are presented some original algorithms, applications and programmed realization.

Keywords: tropical linear algebra; idempotent semirings; matrix Bellman equations; universal algorithms

Acknowledgements: The work is partially supported by the Russian Fund for Basic Research (projects no. 11-01-93106_a, 12-01-00886_a), grant EPSRC RRAH15735.

For citation: Litvinov G.L., Rodionov A.Ya., Sergeev C.N., Sobolevskiy A.N. Universal'nye algoritmy resheniya diskretnyh statsionarnyh uravnenij Bellmana [Universal algorithms for solving discrete stationary Bellman equations]. *Vestnik rossiyskikh universitetov. Matematika – Russian Universities Reports. Mathematics*, 2019, vol. 24, no. 128, pp. 393–431. DOI 10.20310/2686-9667-2019-24-128-393-431. (In Russian, Abstr. in Engl.)

Введение

Вычислительные алгоритмы строятся на основе некоторых простых операций. Эти операции манипулируют данными, которые описывают «числа». Эти «числа» являются элементами «числового домена» (numerical domain), то есть такого математического объекта, как поле вещественных чисел, кольца целых чисел, различные полукольца и т. д.

В практических задачах числовые домены заменяются их компьютерными представлениями, т. е. элементами конкретных конечных моделей этих доменов. Примерами моделей, которые удобно использовать для компьютерного представления вещественных чисел, являются различные модификации арифметики с плавающей запятой, приближенная арифметика рациональных чисел [37], интервальная арифметика и другие. Отличие между математическими объектами («идеальными» числами) и их конечными моделями (машинными представлениями) выражается в вычислительных ошибках (например, при округлении).

Алгоритм называется *универсальным*, если он не зависит от конкретного числового домена и/или его компьютерного представления [32,33,36,40]. Типичным примером универсального алгоритма является вычисление скалярного произведения (x, y) векторов $x = (x_1, \dots, x_n)$ и $y = (y_1, \dots, y_n)$ по формуле $(x, y) = x_1y_1 + \dots + x_ny_n$. Этот алгоритм (формула) не зависит от конкретного домена и его машинной реализации; формула имеет смысл для любого полукольца. Очевидно, что один алгоритм может быть более универсальным, чем другой. Например, простейшая формула Ньютона–Котеса (формула прямоугольников) является самым универсальным алгоритмом численного интегрирования. В частности, эта формула верна также для идемпотентного интегрирования (т. е. над любым идемпотентным полукольцом, см., например, [6,31]). Другие формулы интегрирования (например, формула трапеций или формула Симпсона) не зависят от машинной арифметики и могут использоваться (например, в итерационной форме) для вычислений с произвольной точностью. С другой стороны, алгоритмы, основанные на формулах Гаусса–Якоби, созданы для вычислений с заданной точностью: они содержат константы (коэффициенты этих формул), определенные с конечной точностью (конечно, алгоритмы такого типа можно сделать более универсальными, включив процедуры вычисления констант, однако, это влечет усложнение алгоритмов).

Современные достижения в математике и разработке программного обеспечения позволяют рассматривать численные алгоритмы и их классификацию с новой точки зрения. Обычные численные алгоритмы ориентированы на программную (или аппаратную) реализацию на основе арифметики с плавающей или фиксированной запятой. Тем не менее, часто желательно выполнять вычисления с изменяющейся (и произвольной)

точностью. Для этого от алгоритма требуется независимость от точности вычислений и от особенностей машинного представления чисел. На самом деле многие алгоритмы также не зависят не только от машинного представления чисел, но и от конкретных математических (алгебраических) операций над данными. В этом случае операции сами по себе могут рассматриваться как переменные. Такие алгоритмы реализуются в виде *обобщенных программ*, написанных на основе абстрактных типов данных, которые определяются пользователем в дополнение ко встроенным в язык типам данных. Соответствующие программные средства появились сначала в языке Симула-67, но современные объектно-ориентированные языки (такие как C++, см., например, [41, 47]) более удобны для обобщенного программирования. Алгоритмы компьютерной алгебры, используемые в системах Mathematica, Maple, REDUCE и других, также высоко универсальны.

Другой формой алгоритмов являются итерационные алгоритмы для решения дифференциальных уравнений (например, методы Эйлера, Эйлера–Коши, Рунге–Кутта, Адамса, несколько вариантов метода разностных приближений и подобные), методы вычисления элементарных и некоторых специальных функций, основанных на разложении в ряды Тейлора и на непрерывные дроби (приближения Паде). Эти алгоритмы не зависят от машинного представления чисел.

Понятие обобщенных программ было введено многими авторами: например, в [30] такие программы называются «программными схемами». В настоящей статье мы обсудим универсальные алгоритмы, реализованные в виде обобщенных программ, и их особые возможности. Эта статья тесно связана с работами [8, 31–33, 35, 36, 40], в которых определяется понятие универсального алгоритма, и обсуждается программная и аппаратная реализации таких алгоритмов в связи с задачами идемпотентной математики, см., например, [6, 39, 42, 53, 54].

Так называемый идемпотентный принцип соответствия (см. [32, 33]), связывающий идемпотентную математику с обычной математикой над полями, будет рассмотрен ниже. В двух словах, существует соответствие между интересными, полезными и важными конструкциями и результатами над полем вещественных (или комплексных) чисел и похожими конструкциями над идемпотентыми полукольцами. Это соответствие может быть сформулировано в духе хорошо известного *принципа соответствия Н. Бора* в квантовой механике: на самом деле, оба принципа тесно связаны (см. [31–33]). В известном смысле, традиционную математику над числовыми полями можно рассматривать как «квантовую» теорию, в то время как идемпотентную математику можно рассматривать как «классическую» тень (или двойник) традиционной. Важно, что идемпотентный принцип соответствия верен для алгоритмов, компьютерных программ и их аппаратных реализаций.

В квантовой механике *принцип суперпозиции* означает, что уравнение Шрёдингера (основное уравнение теории) линейно. Подобным образом, в идемпотентной математике принцип суперпозиции, сформулированный В. П. Масловым, означает, что некоторые важные и основные задачи и уравнения, которые являются нелинейными в обычном смысле (например, уравнение Гамильтона–Якоби, одно из основных уравнений классической механики, которое также появляется во многих задачах оптимизации, или уравнение Беллмана и его разновидности и обобщения), могут рассматриваться как

линейные над подходящими идемпотентными полукольцами, см. [4, 5, 31].

Заметим, что численные алгоритмы для бесконечномерных линейных задач над идемпотентными полукольцами (например, идемпотентное интегрирование, интегральные операторы и преобразования, уравнения Гамильтона–Якоби и обобщенные уравнения Беллмана) имеют дело с соответствующими конечномерными приближениями. Следовательно, идемпотентная линейная алгебра является основой идемпотентного численного анализа и, в особенности, *теории дискретной оптимизации.*

Б. А. Карре [18, 19] (см. также [24–26]) использовал идемпотентную линейную алгебру, чтобы показать, что различные задачи оптимизации для конечных графов могут быть сформулированы единым образом и сведены к решению дискретных матричных уравнений Беллмана, то есть некоторых систем линейных алгебраических уравнений над идемпотентными полукольцами. Он также обобщил основные алгоритмы вычислительной линейной алгебры на идемпотентный случай и показал, что некоторые из них совпадают с алгоритмами, разработанными ранее для решения задач оптимизации. Например, метод Беллмана решения задачи нахождения кратчайшего пути соответствует разновидности метода Якоби решения системы линейных уравнений, тогда как алгоритм Форда соответствует методу Гаусса–Зейделя. Уравнения Беллмана являются основной темой данной статьи, а идеи Карре используются для получения новых универсальных алгоритмов. Подчеркнем, что универсальные алгоритмы, описанные в данной статье, могут быть интерпретированы как демонстрация принципа идемпотентной суперпозиции и идемпотентного принципа соответствия между алгоритмами линейной алгебры и соответствующими задачами оптимизации на графах.

Заметим, что многие алгоритмы решения матричных уравнений Беллмана можно найти в [8, 9, 13, 18, 19, 21, 24, 26, 35, 36, 48]. Другие задачи тропической линейной алгебры рассмотрены, например, в [16].

Также мы вкратце обсудим интервальный анализ над идемпотентными и положительными полукольцами. Идемпотентный интервальный анализ возник в работах [3, 10, 39], где он применялся к матричному уравнению Беллмана. Позднее различные задачи, возникающие в интервальной идемпотентной линейной алгебре, были рассмотрены, например, в работах [20, 22, 28, 44, 45]. Важно заметить, что интервалы над идемпотентными полукольцами образуют новое идемпотентное полукольцо. Следовательно, универсальные алгоритмы могут применяться к элементам этого нового полукольца и порождают интервальное расширение исходных алгоритмов.

Данная статья посвящена программным реализациям универсальных алгоритмов решения матричных уравнений Беллмана над полукольцами. Раздел 1. содержит введение в математику полуколец и, особенно, в тропическую (идемпотентную) математику. В разделе 2. мы описываем несколько хорошо известных и новых универсальных алгоритмов линейной алгебры над полукольцами, связанных с дискретным матричным уравнением Беллмана и алгебраической задачей о кратчайшем пути. Эти алгоритмы тесно связаны с их прототипами в традиционной линейной алгебре, описанными, например, в знаменитой книге Дж. Голуба и Ч. Ван Лоуна [1], которая служит основным источником таких прототипов. Следуя стилю [1], мы записываем их на языке MATLAB. Опыт программной реализации универсальных алгоритмов и дальнейшие перспективы их развития также рассматриваются.

1. Математика полуколец

1.1. Основные определения

Широкий класс универсальных алгоритмов связан с понятием полукольца. Напомним соответствующее определение (см., например, [23]). Рассмотрим множество S, снабженное двумя ассоциативными операциями: *сложением* $\oplus$ и *умножением* $\odot$, такими, что сложение коммутативно, умножение дистрибутивно относительно сложения с обеих сторон, $\mathbf{0}$ (соответственно $\mathbf{1}$) нейтральный элемент по сложению (соответственно, по умножению), $\mathbf{0}\odot\mathbf{x}=\mathbf{x}\odot\mathbf{0}=\mathbf{0}$ для всех $x\in S$, и $\mathbf{0}\neq\mathbf{1}$. Пусть полукольцо S частично упорядочено таким отношением $\preceq$, что $\mathbf{0}$ — наименьший элемент, и из неравенства $x\preceq y$ следует, что $x\oplus z\preceq y\oplus z$, $x\odot z\preceq y\odot z$ и $z\odot x\preceq z\odot y$ для всех $x,y,z\in S$; в этом случае полукольцо S называется *положительным* (см., например, [23]).

Полукольцо S называется *полуполем*, если каждый его ненулевой элемент обратим.

Полукольцо S называется *идемпотентным*, если $x\oplus x=x$ для всех $x\in S$. В этом случае сложение $\oplus$ задает *канонический частичный порядок* $\preceq$ на полукольце S по правилу: $x\preceq y$ тогда и только тогда, когда $x\oplus y=y$. Любое идемпотентное кольцо положительно относительно этого порядка. Заметим также, что $x\oplus y=\sup\{x,y\}$ относительно канонического порядка. Впоследствии мы полагаем все идемпотентные кольца упорядоченными канонически.

Мы говорим, что положительное (например, идемпотентное) полукольцо S *полно*, если оно полно как упорядоченное множество. Это означает, что для каждого подмножества $T\subset S$ существуют элементы $\sup T\in S$ и $\inf T\in S$.

Самые известные и важные примеры положительных полуколец — это «числовые» полукольца, состоящие из (подмножества) вещественных чисел, упорядоченных обычным линейным порядком $\leqslant$ на множестве $\mathbf{R}$: полукольцо $\mathbf{R}_+$ с обычными операциями $\oplus=+$, $\odot=\cdot$ и нейтральными элементами $\mathbf{0}=\mathbf{0}$, $\mathbf{1}=\mathbf{1}$, полукольцо $\mathbf{R}_{\max}=\mathbf{R}\cup\{-\infty\}$ с операциями $\oplus=\max$, $\odot=+$ и нейтральными элементами $\mathbf{0}=-\infty$, $\mathbf{1}=\mathbf{0}$, полукольцо $\widehat{\mathbf{R}}_{\max}=\mathbf{R}_{\max}\cup\{\infty\}$, где $x\preceq\infty$, $x\oplus\infty=\infty$ для всех x, $x\odot\infty=\infty\odot x=\infty$, если $x\neq 0$, и $\mathbf{0}\odot\infty=\infty\odot\mathbf{0}$ и полукольцо $S^{[a,b]}_{\max,\min}=[a,b]$, где $-\infty\leqslant a<b\leqslant+\infty$ с операциями $\oplus=\max$, $\odot=\min$ и нейтральными элементами $\mathbf{0}=\mathbf{a}$, $\mathbf{1}=\mathbf{b}$. Полукольца $\mathbf{R}_{\max}$, $\widehat{\mathbf{R}}_{\max}$, и $S^{[a,b]}_{\max,\min}=[a,b]$ идемпотентны. Полукольца $\widehat{\mathbf{R}}_{\max}$, $S^{[a,b]}_{\max,\min}$, $\widehat{\mathbf{R}}_+=\mathbf{R}_+\bigcup\{\infty\}$ полны как упорядоченные множества. Вспомним, что каждое частично упорядоченное множество может быть вложено в свое пополнение (минимальное полное множество, содержащее исходное). Полукольцо $\mathbf{R}_{\min}=\mathbf{R}\bigcup\{\infty\}$ с операциями $\oplus=\min$ и $\odot=+$ и нейтральными элементами $\mathbf{0}=\infty$, $\mathbf{1}=0$ изоморфно полукольцу $\mathbf{R}_{\max}$.

Полукольцо $\mathbf{R}_{\max}$ называют *алгеброй макс-плюс*. Полуполя $\mathbf{R}_{\max}$ и $\mathbf{R}_{\min}$ также называют *тропическими алгебрами*. Термин «тропический» впервые появился в [51] для целочисленной версии алгебры макс-плюс, см. также [27, 42, 54].

Обозначим $\mathrm{Mat}_{mn}(S)$ множество всех матриц $A=(a_{ij})$ с m строками и с n столбцами, элементы которых принадлежат полукольцу S. Сумма $A\oplus B$ матриц $A,B\in\mathrm{Mat}_{mn}(S)$ и произведение AB матриц $A\in\mathrm{Mat}_{lm}(S)$ и $B\in\mathrm{Mat}_{mn}(S)$ определяются в соответ-

ствии с обычными правилами линейной алгебры: $A \oplus B = (a_{ij} \oplus b_{ij}) \in \mathrm{Mat}_{mn}(S)$ и

$$AB = \left(\bigoplus_{k=1}^{m} a_{ij} \odot b_{kj} \right) \in \mathrm{Mat}_{ln}(S),$$

где $A \in \mathrm{Mat}_{lm}(S)$ и $B \in \mathrm{Mat}_{mn}(S)$.

Если полукольцо S положительно, то множество $\mathrm{Mat}_{mn}(S)$ упорядочено посредством отношения $A = (a_{ij}) \preceq B = (b_{ij})$ тогда и только тогда, когда $a_{ij} \preceq b_{ij}$ в S для всех $2 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$.

Умножение матриц согласуется с порядком $\preceq$ в следующем смысле: если $A, A' \in \mathrm{Mat}_{lm}(S)$, $B, B' \in \mathrm{Mat}_{mn}(S)$ и $A \preceq A'$, $B \preceq B'$, то $AB \preceq A'B'$ в $\mathrm{Mat}_{ln}(S)$. Более того, множество $\mathrm{Mat}_{nn}(S)$ квадратных матриц $(n \times n)$ над (положительным идемпотентным) полукольцом S образует [положительное идемпотентное] полукольцо с нулевым элементом $O = (o_{ij})$, где $o_{ij} = \mathbf{0}$, $1 \leqslant i, j \leqslant n$, и единичным элементом $I = (\delta_{ij})$, где $\delta_{ij} = \mathbf{1}$, если $i = j$, и $\delta_{ij} = \mathbf{0}$ в противном случае.

Множество Mat_{nn} — типичный пример некоммутативного полукольца (при $n > 1$).

1.2. Операция замыкания

Пусть положительное полукольцо S снабжено частичной унарной *операцией замыкания* $*$ такой, что из $a \preceq b$ следует, что $a^* \preceq b^*$ и $a^* = \mathbf{1} \oplus (\mathbf{a}^* \odot \mathbf{a}) = \mathbf{1} \oplus (\mathbf{a} \odot \mathbf{a}^*)$ во всей ее области определения. В частности, $\mathbf{0}^* = \mathbf{1}$ по определению.

Из этих аксиом следует, что $a^* = \mathbf{1} \oplus \mathbf{a} \oplus \mathbf{a^2} \oplus \cdots \oplus (\mathbf{a}^* \odot \mathbf{a^n})$, если $n \geqslant 1$. Таким образом, x^* можно рассматривать как «регуляризованную сумму» ряда

$$a^* = \mathbf{1} \oplus \mathbf{a} \oplus \mathbf{a^2} \oplus \dots .$$

В положительном полукольце, при условии, что оно замкнуто относительно взятия ограниченных упорядоченных верхних граней и операций $\oplus$ и $\odot$, дистрибутивных относительно таких верхних граней, мы можем определить

$$a^* := \sup_{k \geqslant 0} \mathbf{1} \oplus \mathbf{a} \oplus \dots \oplus \mathbf{a^k}, \tag{1.1}$$

если последовательность правых частей ограничена. В этом случае a^* — **наименьшее решение** уравнений $x = ax \oplus \mathbf{1}$ и $x = xa \oplus \mathbf{1}$, и a^*b — наименьшее решение уравнений $x = ax \oplus b$ и $x = xa \oplus b$. Если S полно, то замыкание очевидным образом определено для любого элемента $x \in S$.

В случае идемпотентного сложения (1.1) становится особенно приятным:

$$a^* = \bigoplus_{i \geqslant 0} a^i = \sup_{i \geqslant 0} a^i. \tag{1.2}$$

В числовых полукольцах операция замыкания $*$ обычно реализуется очень просто: $x^* = (1 - x)^{-1}$, если $x < 1$ в $\mathbf{R}_+$, или в $\widehat{\mathbf{R}}_+$, и $x^* = \infty$, если $x \geqslant 1$ в $\widehat{\mathbf{R}}_+$; $x^* = \mathbf{1}$, если $x \preceq \mathbf{1}$ в $\mathbf{R}_{\max}$ и $\widehat{\mathbf{R}}_{\max}$, $x^* = \infty$, если $x \succ \mathbf{1}$ в $\widehat{\mathbf{R}}_{\max}$, $x^* = \mathbf{1}$ для всех x в ${S_{\max,\min}}^{[a,b]}$. Во всех остальных случаях x^* не определено. Операция замыкания в матричных полукольцах над положительным полукольцом S может быть определена

по индукции: во-первых, имеем $A^* = (a_{11})^* = (a_{11}^*)$ в $\mathrm{Mat}_{11}(S)$. Во-вторых, для любого целого числа $n > 1$ и любой матрицы

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

где $A_{11} \in \mathrm{Mat}_{kk}(S)$, $A_{12} \in \mathrm{Mat}_{k\,n-k}(S)$, $A_{21} \in \mathrm{Mat}_{n-k\,k}(S)$, $A_{22} \in \mathrm{Mat}_{n-k\,n-k}(S)$, $1 \leqslant k \leqslant n$, определим

$$A^* = \begin{pmatrix} A_{11}^* \oplus A_{11}^* A_{12} D^* A_{21} A_{11}^* & A_{11}^* A_{12} D^* \\ D^* A_{21} A_{11}^* & D^* \end{pmatrix}, \tag{1.3}$$

где $D = A_{22} \oplus A_{21} A_{11}^* A_{12}$. Можно доказать, что из определения A^* следует, что равенства $A^* = A^* A \oplus I = A A^* \oplus I$ выполняются, и, таким образом, A^* является «регуляризованной суммой» ряда $I \oplus A \oplus A^2 \oplus \ldots$. Более того, в случае когда A^* определено как наименьшее решение уравнения $A^* = A^* A \oplus I$ и $A^* = A A^* \oplus I$, можно показать, что оно удовлетворяет уравнению (1.3).

Заметим, что это рекуррентное соотношение совпадает с формулами эскалаторного метода обращения матриц в обычной линейной алгебре над полями вещественных и комплексных чисел с точностью до используемых алгебраических операций. Следовательно, этот алгоритм матричного замыкания требует полиномиального от n числа операций, подробнее см. ниже.

Пусть S положительное полукольцо. *Матричное (дискретное, стационарное) уравнение Беллмана* имеет вид

$$X = AX \oplus B, \tag{1.4}$$

где $A \in \mathrm{Mat}_{nn}(S)$, $X, B \in \mathrm{Mat}_{ns}(S)$, а матрица X неизвестна. Пусть A^* замыкание матрицы A. Из тождества $A^* = A^* A \oplus I$ следует, что матрица $A^* B$ удовлетворяет этому уравнению. Также как и в скалярном случае, можно показать (при некотором дополнительном предположении) , что для положительных полуколец, если матрица A^* определена как в уравнении (1.1), то $A^* B$ является наименьшим во множестве решений уравнения (1.4) относительно частично порядка на множестве $\mathrm{Mat}_{ns}(S)$. Вспомним, что в идемпотентном случае

$$A^* = \bigoplus_{i \geqslant 0} A^i = \sup_{i \geqslant 0} A^i. \tag{1.5}$$

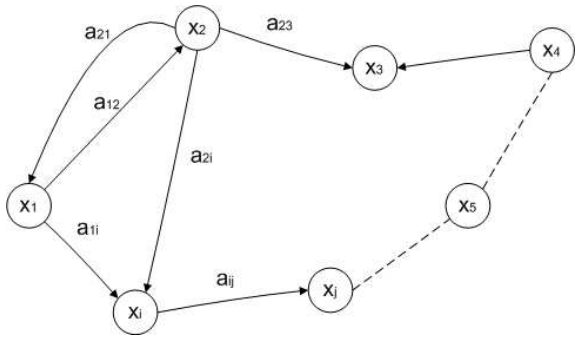
Рассмотрим также случай, когда матрица $A = (a_{ij})$ размерности $n \times n$ является *строго верхнетреугольной* (такой, что $a_{ij} = \mathbf{0}$ при $i \geqslant j$) или *строго нижнетреугольной* (такой, что $a_{ij} = \mathbf{0}$ при $i \leqslant j$). В этом случае $A^n = 0$, нулевой матрице, и можно показать с помощью итераций $X = AX \oplus I$, что это уравнение имеет единственное решение, а именно

$$A^* = I \oplus A \oplus \ldots \oplus A^{n-1}. \tag{1.6}$$

Интересно, что в случае алгебры макс-плюс формула (1.6) работает для более широкого класса матриц. Ряд (1.5) сходится там тогда и только тогда, когда он может быть усечен до (1.6). Это тесно связано с интерпретацией матрицы A^* как «матрицы оптимальных путей» (см. ниже).

1.3. Взвешенные ориентированные графы и матрицы над полукольцами

Пусть S — полукольцо с нулем $\mathbf{0}$ и единицей $\mathbf{1}$. Хорошо известно, что любая квадратная матрица $A = (a_{ij}) \in \mathrm{Mat}_{nn}(S)$ задает *взвешенный ориентированный граф*. Эта геометрическая конструкция включает три вида объектов: множество X из n элементов $x_1, \dots, x_n$, называемых *вершинами,* множество Γ всех упорядоченных пар (x_i, x_j) таких, что $a_{ij} \neq \mathbf{0}$, называется *дугами*, и отображение $A\colon \Gamma \to S$ такое, что $A(x_i, x_j) = a_{ij}$. Элементы a_{ij} из полукольца S называются *весами* дуг. Наоборот, любой взвешенный ориентированный граф с n вершинами задает некоторую (единственную) матрицу $A \in \mathrm{Mat}_{nn}(S)$.



Это определение допускает, что некоторые пары вершин могут быть не соединены, если соответствующий элемент матрицы A равен $\mathbf{0}$ и некоторые дуги могут быть петлями, если у матрицы A есть ненулевые элементы на главной диагонали.

Вспомним, что последовательность вершин вида:

$$p = (y_0, y_1, \dots, y_k),$$

где $k \geqslant 0$ и $(y_i, y_{i+1}) \in \Gamma$, $i = 0, \dots, k-1$, называется *путем* длины k, соединяющим вершину y_0 с вершиной y_k. Обозначим множество всех таких путей $P_k(y_0, y_k)$. Вес $A(p)$ пути $p \in P_k(y_0, y_k)$ определяется как произведение весов дуг, соединяющих последовательные вершины пути:

$$A(p) = A(y_0, y_1) \odot \cdots \odot A(y_{k-1}, y_k).$$

По определению, вес «пути» $p \in P_0(x_i, x_j)$ длины $k = 0$ равен $\mathbf{1}$, если $i = j$, и $\mathbf{0}$ в противном случае.

Для каждой матрицы $A \in \mathrm{Mat}_{nn}(S)$ определим $A^0 = I = (\delta_{ij})$ (где $\delta_{ij} = \mathbf{1}$, если $i = j$, и $\delta_{ij} = \mathbf{0}$ в противном случае) и $A^k = AA^{k-1}$, $k \geqslant 1$. Пусть $a_{ij}^{[k]}$ — (i,j)-й элемент матрицы A^k. Легко проверяется, что

$$a_{ij}^{[k]} = \bigoplus_{\substack{i_0 = i,\, i_k = j \\ 1 \leqslant i_1, \dots, i_{k-1} \leqslant n}} a_{i_0 i_1} \odot \cdots \odot a_{i_{k-1} i_k}.$$

Таким образом, $a_{ij}^{[k]}$ — точная верхняя грань множества весов, отвечающих всем путям длины k, соединяющим вершину $x_{i_0} = x_i$ с вершиной $x_{i_k} = x_j$.

Пусть матрица A^* определена в соответствии с формулой (1.5). Обозначим элементы матрицы A^* как a^*_{ij}, $i,j=1,\dots,n$; тогда

$$a^*_{ij} = \bigoplus_{0\leqslant k<\infty} \bigoplus_{p\in P_k(x_i,x_j)} A(p).$$

Матрица A^* является решением известного алгебраического обобщения *задачи на нахождение оптимального пути*: для каждой пары (x_i,x_j) требуется вычислить точную верхнюю грань весов всех путей (произвольной длины), соединяющих вершину x_i с вершиной x_j. Операция замыкания в полукольце матриц хорошо изучена (см., например, [2,6,13,16,18,19,21,23,25,26,39] и ссылки в них).

Пример 1.1 (Задача о кратчайшем пути). Пусть $S=\mathbf{R}_{\min}$, т. е. веса являются вещественными числами. В этом случае

$$A(p) = A(y_0,y_1) + A(y_1,y_2) + \dots + A(y_{k-1},y_k).$$

Если элемент a_{ij} задает длину дуги (x_i,x_j) в некоторой метрике, тогда a^*_{ij} — длина кратчайшего пути, соединяющего x_i с x_j.

Пример 1.2 (Задача о пути максимальной ширины). Пусть $S=\mathbf{R}\cup\{\mathbf{0},\mathbf{1}\}$ с $\oplus=\max$, $\odot=\min$. Тогда

$$a^*_{ij} = \max_{p\in\bigcup\limits_{k\geqslant 1} P_k(x_i,x_j)} A(p),$$

$$A(p) = \min(A(y_0,y_1),\dots,A(y_{k-1},y_k)).$$

Если элемент a_{ij} задает «ширину» дуги (x_i,x_j), тогда ширина пути p определяется как наименьшая ширина дуги, принадлежащей этому пути, а элемент a^*_{ij} является максимальной шириной всех путей, соединяющих x_i с x_j.

Пример 1.3 (Простая задача динамического программирования). Пусть $S=\mathbf{R}_{\max}$, и положим, что a_{ij} задает *выплаты* при переходе от x_i к x_j. Определим вектор $B=(b_i)\in \mathrm{Mat}_{n1}(\mathbf{R}_{\max})$, элементы которого b_i задают *терминальные выплаты*, соответствующие выходу из графа в узле x_i. Конечно, отрицательные выплаты (то есть убытки) также допускаются. Пусть m — итоговая выплата, соответствующая пути $p\in P_k(x_i,x_j)$, т. е.

$$m = A(p) + b_j.$$

Тогда легко проверить, что наибольшая выплата, которая может быть достигнута на путях длины k, начинающихся в узле x_i, равна $(A^kB)_i$, и наибольшая выплата, достижимая без ограничений на длину пути, равна $(A^*B)_i$. Заметим, что если длина пути неограничена, то наибольшая выплата может быть равна бесконечности, как и соответствующее значение $(A^*B)_i$.

Пример 1.4 (Задача обращения матрицы). Заметим, что в формулах этого раздела мы используем дистрибутивность умножения $\odot$ относительно сложения $\oplus$, но не

используем аксиому идемпотентности. Таким образом, алгебраическое обобщение задачи об оптимальном пути может быть сформулировано также и для неидемпотентного полукольца S (см., например, [48]). Например, если $S = \mathbf{R}$, то

$$A^* = I + A + A^2 + \cdots = (I - A)^{-1}.$$

Если $\|A\| > 1$, но матрица обратима, то это выражение определяет регуляризованную сумму расходящегося матричного степенного ряда $\sum_{i \geqslant 0} A^i$.

Подчеркнем, что эта связь между матричной операцией замыкания и решениями уравнения Беллмана порождает несколько различных алгоритмов для вычисления матричного замыкания. Все эти алгоритмы являются адаптациями хорошо известных алгоритмов традиционной вычислительной линейной алгебры, таких как метод исключения Гаусса–Жордана, различные итерационные и эскалаторные схемы и т. д. Это особый случай идемпотентного принципа суперпозиции (см. ниже).

На самом деле, теория дискретного стационарного уравнения Беллмана может быть развита с использованием равенства $A^* = AA^* \oplus I$ как дополнительной аксиомы без какой-либо существенной интерпретации (так называемые *замкнутые полукольца*, см., например, [23, 30, 48]). Такая теория может быть основана на следующих равенствах, которые в случае идемпотентных полуколец можно проверить, используя связь с задачей на нахождение оптимального пути. Эти равенства также хорошо известны и в случае вещественных чисел с обычной арифметикой:

$$\begin{aligned} (A \oplus B)^* &= (A^* B)^* A^*, \\ (AB)^* A &= A(BA)^*. \end{aligned} \tag{1.1}$$

1.4. Интервальный анализ над положительными полукольцами

Традиционный интервальный анализ — это нетривиальная и популярная математическая область, см., например, [12, 22, 29, 43, 46]. «Идемпотентная» версия интервального анализа (и кроме того, интервального анализа над положительными полукольцами) появилась в [3, 10, 39]. Довольно много работ по этой теме появились позднее, см., например, [20, 22, 28, 44, 45]. Интервальный анализ над положительным полукольцом $\mathbf{R}_+$ обсуждается в [15].

Пусть множество S частично упорядочено отношением $\preceq$. *Замкнутый интервал* в S – это подмножество вида $\mathbf{x} = [\underline{x}, \overline{x}] = \{ x \in S \mid \underline{x} \preceq x \preceq \overline{x} \}$, где элементы $\underline{x} \preceq \overline{x}$ называются *нижней* и *верхней границей* интервала $\mathbf{x}$. Порядок $\preceq$ порождает частичное упорядочение на множестве всех замкнутых интервалов в S: $\mathbf{x} \preceq \mathbf{y}$ тогда и только тогда, когда $\underline{x} \preceq \underline{y}$ и $\overline{x} \preceq \overline{y}$.

Слабым интервальным расширением $I(S)$ положительного полукольца S называется множество всех замкнутых интервалов в S, снабженное операциями $\oplus$ и $\odot$, определенными следующим образом: $\mathbf{x} \oplus \mathbf{y} = [\underline{x} \oplus \underline{y}, \overline{x} \oplus \overline{y}]$, $\mathbf{x} \odot \mathbf{y} = [\underline{x} \odot \underline{y}, \overline{x} \odot \overline{y}]$, и частичным порядком, порожденным порядком на S. Операция замыкания в $I(S)$ определяется как $\mathbf{x}^* = [\underline{x}^*, \overline{x}^*]$. Существуют другие интервальные расширения (в том числе, так называемые сильные интервальные расширения [39]), но слабое расширение более удобно.

Расширение $I(S)$ положительное; $I(S)$ идемпотентно, если S — идемпотентное полукольцо. Универсальный алгоритм над S может применяться к $I(S)$, и мы получим интервальную версию исходного алгоритма. Обычно обе версии имеют одинаковую сложность. Для дискретного стационарного уравнения Беллмана и соответствующей оптимизационной задачи на графах интервальный анализ подробно рассматривается в [3, 39]. Другие задачи идемпотентной линейной алгебры были рассмотрены в [20, 22, 28, 44, 45].

Идемпотентная математика оказывается проще ее традиционного аналога. Например, в традиционной интервальной арифметике перемножение интервалов не дистрибутивно относительно сложения, тогда как в идемпотентной интервальной арифметике эта дистрибутивность сохраняется. Более того, в традиционном интервальном анализе множество всех квадратных интервальных матриц заданного порядка не образует даже полугруппы по матричному умножению: эта операция не ассоциативна, поскольку дистрибутивность теряется в традиционной интервальной арифметике. Наоборот, в идемпотентном (и положительном) случае ассоциативность сохраняется. Наконец, в обычном интервальном анализе некоторые задачи линейной алгебры, такие как решение линейных систем интервальных уравнений, может быть очень сложным (точнее говоря, они NP-полные, см. [29] и ссылки там же). В [3, 39] показано, что в идемпотентном случае решение интервальных линейных систем требует полиномиального количества операций (так же как и для обычного метода исключения Гаусса). Два свойства, которые делают идемпотентную интервальную арифметику такой простой, — это монотонность арифметических операций и положительность всех элементов идемпотентного полукольца.

Отметим, что интервальные оценки в идемпотентной математике обычно являются точными. В традиционной теории такие оценки, как правило, слишком пессимистические.

1.5. Идемпотентный принцип соответствия

Существует нетривиальная аналогия между математикой полуколец и квантовой механикой. Например, поле вещественных чисел можно рассматривать как «квантовый объект» относительно идемпотентных полуколец, которые рассматриваются как «классические» и «полуклассические» объекты относительно поля вещественных чисел.

Пусть $\mathbf{R}$ — поле вещественных чисел, а $\mathbf{R}_+$ — подмножество всех неотрицательных чисел. Рассмотрим следующую замену переменных:

$$u \mapsto w = h \ln u,$$

где $u \in \mathbf{R}_+ \setminus \{\mathbf{0}\}$, $h > 0$; таким образом, $u = e^{w/h}$, $w \in \mathbf{R}$. Обозначим через $\mathbf{0}$ дополнительный элемент $-\infty$ и через S расширенную вещественную прямую $\mathbf{R} \cup \{\mathbf{0}\}$. Эта замена переменных имеет естественное расширение D_h на все S с $D_h(0) = \mathbf{0}$; также мы положим $D_h(1) = 0 = \mathbf{1}$.

Обозначим S_h множество S, снабженное двумя операциями $\oplus_h$ (обобщенное сложение) и $\odot_h$ (обобщенное умножение), такое что D_h — гомоморфизм из $\{\mathbf{R}_+, +, \cdot\}$ в

$\{S, \oplus_h, \odot_h\}$. Это означает, что

$$D_h(u_1 + u_2) = D_h(u_1) \oplus_h D_h(u_2), \quad D_h(u_1 \cdot u_2) = D_h(u_1) \odot_h D_h(u_2),$$

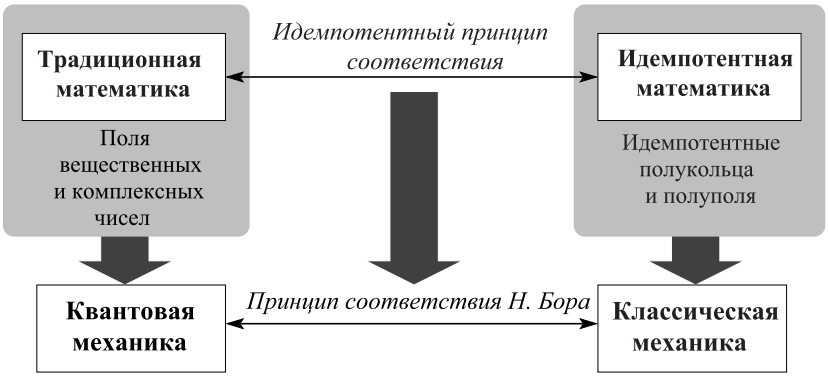
то есть $w_1 \odot_h w_2 = w_1 + w_2$ и $w_1 \oplus_h w_2 = h \ln(e^{w_1/h} + e^{w_2/h})$. Легко показать, что $w_1 \oplus_h w_2 \to \max\{w_1, w_2\}$ при $h \to 0$.

Алгебраически $\mathbf{R}_+$ и S_h изоморфны, поэтому мы получаем $\mathbf{R}_{\max}$ как результат предельной деформации $\mathbf{R}_+$. Мы подчеркиваем очевидную аналогию с процедурой деквантования, где h — аналог постоянной Планка. Таким образом, $\mathbf{R}_+$ (или $\mathbf{R}$) играет роль «квантового объекта», а $\mathbf{R}_{\max}$ — «классического» или «полуклассического» объекта, который получается как результат *деквантования* квантового объекта $\mathbf{R}_+$. В случае $\mathbf{R}_{\min}$ соответствующая процедура деквантования порождается заменой переменных $u \mapsto w = -h \ln u$.

В случае поля вещественных или комплексных чисел, идемпотентное деквантование в полукольцо $\mathbf{R}_{\max}$ (или $\mathbf{R}_{\min}$) определяется с помощью композиции отображения $x \mapsto |x|$ и деформации, описанной выше.

В то же время идемпотентное деквантование играет важную роль в методологии и в философии идемпотентной математики как обоснование возможности переноса с основного поля (как правило, вещественных или комплексных чисел) на идемпотентное полукольцо многих математических понятий, конструкций и результатов, см. [31, 33]. Идемпотентное деквантование приводит к следующей формулировке *идемпотентного принципа соответствия* [31, 33]:

> *Существует эвристическое соответствие между интересными, полезными и важными результатами над полем вещественных (или комплексных) чисел и похожими конструкциями и результатами над идемпотентными полукольцами в духе принципа соответствия Н. Бора в квантовой механике.*



Идемпотентная математика может рассматриваться как «классическая тень (или двойник)» традиционной математики над полями. Систематическое приложение этого принципа соответствия ведет ко множеству теоретических и прикладных результатов, см., например, [31,33,36,39,42,53,54]. Связь с квантовой физикой подробно обсуждается, например, в [31].

Теперь мы обсудим идею о том, как идемпотентный принцип соответствия мог бы привести к унифицированному подходу к разработке аппаратного обеспечения. Подробнее об этой идее и ее осуществлении см. в [35, 40].

Наиболее важные и давно известные численные алгоритмы имеют множество аппаратных реализаций в виде технических устройств или специальных процессоров. Часто эти устройства могут использоваться как прототипы для новых аппаратных устройств, в результате простой замены обычных арифметических операций на их аналоги в полукольцах (и дополнительные средства для порождения нейтральных элементов $\mathbf{0}$ и $\mathbf{1}$). Разумеется, случай числовых полуколец, состоящих из вещественных чисел (возможно, кроме нейтральных элементов) и полуколец числовых интервалов, является самым простым и естественным. Заметим, что для полуколец (включая $\mathbf{R}_{\max}$ и $\mathbf{R}_{\min}$) операция деления также определена.

Эффективные технические идеи и решения могут быть заимствованы из прототипов и реализованы в новых аппаратных устройвах. Таким образом, принцип соответствия порождает эвристический метод для разработки аппаратных средств. Заметим, что для получения патента необходимо представить так называемую «формулу изобретения», где требуется указать прототип предлагаемого устройства и разницу между устройством и его прототипом.

Рассмотрим (как типичный пример) самый важный и давно известный алгоритм вычисления скалярного произведения двух векторов:

$$(x, y) = x_1y_1 + x_2y_2 + \cdots + x_ny_n. \tag{1.1}$$

Универсальная версия формулы (1.1) для любого полукольца A очевидна:

$$(x, y) = (x_1 \odot y_1) \oplus (x_2 \odot y_2) \oplus \cdots \oplus (x_n \odot y_n). \tag{1.2}$$

В случае $A = \mathbf{R}_{\max}$ эта формула принимает следующий вид:

$$(x, y) = \max\{x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n\}. \tag{1.3}$$

Это вычисление стандартно для многих алгоритмов оптимизации, поэтому полезно создать аппаратное средство для вычисления (1.3). Существует множество различных устройств и патентов для вычисления (1.1), и каждое такое устройство можно использовать в качестве прототипа для создания нового устройства для вычисления (1.3) и даже (1.2). Многие процессоры для перемножения матриц и других алгоритмов линейной алгебры основаны на вычислении скалярного произведения и на соответствующих «элементарных» устройствах. Используя современные технологии, можно создать дешевые специализированные многопроцессорные чипы и систолические массивы элементарных процессоров, реализующих универсальные алгоритмы. См., например, [35, 40, 48], где обсуждаются систолические массивы и вопросы параллельного вычисления в рамках задачи алгебраического пути. В частности, существует систолический массив из $n(n + 1)$ элементарных процессоров, которые выполняют алгоритм исключения Гаусса–Жордана и могут решать задачу алгебраического пути за $5n - 2$ временных шагов.

2. Некоторые универсальные алгоритмы линейной алгебры

В этом разделе мы рассмотрим универсальные алгоритмы вычисления A^* и A^*B. Мы начнем с простых методов: эскалаторного метода и метода исключения Гаусса–Жордана в подразделе 2.1., а потом рассмотрим модификацию этих методов для случая систем Тёплица (подраздел 2.2.). Универсальное LDM-разложение для уравнения Беллмана описано в подразделе 2.3., а потом приводится адаптация этого метода для случаев симметричной и ленточной матрицы (подраздел 2.4.) . Далее обсуждаются две итерационные схемы, а также реализация универсальных алгоритмов (подраздел 2.7.).

Сами алгоритмы будут записаны на языке MATLAB, следуя книге Голуба и ван Лоуна. Это сделано по следующим причинам: 1) желание упростить сравнение алгоритмов с их аналогами, взятых преимущественно из [1]; 2) поскольку язык MATLAB разработан для матричных вычислений и удобен для их записи. Мы не будем формально описывать правила нашего MATLAB-образного языка, предпочитая только выделить следующие важные особенности:

1. Основные арифметические операции: $a \oplus b$, $a \odot b$ и a^*.
2. Эти же операции для векторов следуют правилам MATLAB.
3. Мы используем основные ключевые слова MATLAB: «for», «if» и «end», похожие на ключевые слова других языков программирования (C++, Java, ...).

Дадим несколько примеров универсальных матричных вычислений на нашем языке.

П р и м е р 2.1. $v(1:j) = \alpha^* \odot a(1:j,k)$ означает, что результат умножения (скалярного) первых j компонент k-й колонки матрицы A на замыкание α присваивается первым j компонентам вектора v.

П р и м е р 2.2. $a(i,j) = a(i,j) \oplus a(i,1:n) \odot a(1:n,j)$ означает, что мы добавляем ($\oplus$) к элементу a_{ij} матрицы A результат скалярного умножения (универсального) i-й строки и j-го столбца матрицы A (предполагается, что A – матрица $n \times n$).

П р и м е р 2.3. $a(1:n,k) \odot b(l,1:m)$ означает внешнее произведение k-го столбца матрицы A и l-й строки матрицы B. Элементы матрицы-результата $C = (c_{ij})$ равны $c_{ij} = a_{ik} \odot b_{lj}$ для всех $i = 1, \dots, n$ и $j = 1, \dots, m$.

П р и м е р 2.4. $x(1:n) \odot y(n:-1:1)$ – скалярное произведение вектора x и вектора y, чьи компоненты берутся в обратном порядке: правильное алгебраическое выражение $\bigoplus_{i=1}^{n} x_i \odot y_{n+1-i}$.

П р и м е р 2.5. Следующий цикл вычисляет тот же результат, что и предыдущий пример:

$s = 0$
for $i = 1:n$
$s = s \oplus x(i) \odot x(n+1-i)$
end

2.1. Эскалаторная схема и исключение Гаусса-Жордана

Сначала мы проанализируем базовый эскалаторный метод, основанный на определении замыкания матрицы (1.3). Пусть A — квадратная матрица. Замыкания ее главных подматриц A_k могут быть найдены по индукции, начиная с $A_1^* = (a_{11})^*$, замыкания первого диагонального элемента. В общем случае мы выражаем A_{k+1} как

$$A_{k+1} = \begin{pmatrix} A_k & g_k \\ h_k^T & a_{k+1} \end{pmatrix},$$

предполагая, что мы уже нашли замыкание для A_k. В этой формуле g_k и h_k — столбцы с k элементами и a_{k+1} — скаляр. Мы также выражаем A_{k+1}^* как

$$A_{k+1}^* = \begin{pmatrix} U_k & v_k \\ w_k^T & u_{k+1} \end{pmatrix}.$$

Используя (1.3), мы получаем, что

$$\begin{aligned} u_{k+1} &= (h_k^T A_k^* g_k \oplus a_{k+1})^*, \\ v_k &= A_k^* g_k u_{k+1}, \\ w_k^T &= u_{k+1} h_k^T A_k^*, \\ U_k &= A_k^* g_k u_{k+1} h_k^T A_k^* \oplus A_k^*. \end{aligned} \tag{2.1}$$

Алгоритм, основанный на (2.1), записывается следующим образом.

Алгоритм 1. *Эскалаторный метод вычисления для* A^*

Вход: матрица A размером $n \times n$ с элементами $a(i,j)$,
также используется для хранения результата вычисления
и промежуточных результатов вычисления.

$a(1,1) = (a(1,1))^*$
for $i = 1 : n-1$
$Ag = a(1:i, 1:i) \odot a(1:i, i+1)$
$hA = a(i+1, 1:i) \odot a(1:i, 1:i)$
$a(i+1, i+1) = a(i+1, i+1) \oplus a(i+1, 1:i) \odot Ag(1:i, 1)$
$a(i+1, i+1) = (a(i+1, i+1))^*$
$a(1:i, i+1) = a(i+1, i+1) \odot Ag$
$a(i+1, 1:i) = a(i+1, i+1) \odot hA$
$a(1:i, 1:i) = a(1:i, 1:i) \oplus Ag \odot a(i+1, i+1) \odot hA$
end

В полной аналогии со своим прототипом из линейной алгебры алгоритм требует $n^3 + O(n^2)$ операций сложения $\oplus$, $n^3 + O(n^2)$ операций умножения $\odot$ и n операций взятия алгебраического замыкания. Линейно-алгебраический прототип метода, записанного выше, также называется в литературе [11, 18] *методом окаймления.*

В качестве альтернативы, мы можем получить решение $X = AX \oplus B$ в результате процесса исключения, формальное объяснение которого дано ниже. Если A^* определяется как $\bigoplus_{i \geqslant 0} A^i$ (включая скалярный случай), тогда A^*B наименьшее решение

$X = AX \oplus B$ для всех A и B подходящих размеров. В этом случае, решение, найденное с помощью процесса исключения, совпадает с A^*B.

Для матрицы $A = (a_{ij})$ и вектор-столбцов $x = (x_i)$ и $b = (b_i)$ (ограничимся без потери общности случаем, когда x и b — вектор-столбцы) уравнение Беллмана $x = Ax \oplus b$ может быть записано как

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \oplus \begin{pmatrix} \mathbf{1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}. \tag{2.2}$$

После выражения x_1 через $x_2, \dots, x_n$ из первого уравнения и подстановки этого выражения для x_1 во все другие уравнения от второго до n-го мы получим:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \mathbf{0} & (a_{11})^* a_{12} & \dots & (a_{11})^* a_{1n} \\ \mathbf{0} & a_{22} \oplus (a_{21}(a_{11})^* a_{12}) & \dots & a_{2n} \oplus (a_{21}(a_{11})^* a_{1n}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & a_{n2} \oplus (a_{n1}(a_{11})^* a_{12}) & \dots & a_{nn} \oplus (a_{n1}(a_{11})^* a_{1n}) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$
$$\oplus \begin{pmatrix} (a_{11})^* & \mathbf{0} & \dots & \mathbf{0} \\ a_{21}(a_{11})^* & \mathbf{1} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}(a_{11})^* & \mathbf{0} & \dots & \mathbf{1} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{2.3}$$

Заметим, что нетривиальные элементы в обеих матрицах занимают дополнительные по отношению друг к другу позиции, поэтому во время вычислений обе матрицы могут храниться в одной квадратной матрице $C^{(k)}$. Обозначим ее элементы через $c_{ij}^{(k)}$, где k — число исключенных переменных. После $l-1$ исключения мы имеем

$$\begin{aligned} x_l &= (c_{ll}^{(l-1)})^* b_l, \\ c_{il}^{(l)} &= c_{il}^{(l-1)} (c_{ll}^{(l-1)})^*, \quad i = 1, \dots, l-1, l+1, \dots, n \\ c_{ij}^{(l)} &= c_{ij}^{(l-1)} \oplus c_{il}^{(l-1)} (c_{ll}^{(l-1)})^* c_{lj}^{(l-1)}, \\ i, j &= 1, \dots, l-1, l+1, \dots, n \\ c_{li}^{(l)} &= (c_{ll}^{(l-1)})^* c_{li}^{(l-1)}, \quad i = 1, \dots, l-1, l+1, \dots, n \end{aligned} \tag{2.4}$$

После n исключений имеем $x = C^{(n)} b$. Выбирая в качестве b любой вектор с одной координатой, равной $\mathbf{1}$, а остальными, равными $\mathbf{0}$, получим $C^{(n)} = A^*$. Запишем следующий алгоритм, основанный на рекурсии (2.4).

Алгоритм 2. *Метод исключения Гаусса–Жордана для вычисления* A^*.

Вход: матрица A размеров $n \times n$ с элементами $a(i,j)$,
также используется для хранения окончательного результата
и промежуточные результаты вычислений.

for $i = 1 : n$
$a(i,i) = (a(i,i))^*$

for $k = 1 : n$
if $k \neq i$
$a(k,i) = a(k,i) \odot a(i,i)$
end end
for $k = 1 : n$
for $j = 1 : n$
if $k \neq i$ & $j \neq i$
$a(k,j) = a(k,j) \oplus a(k,i) \odot a(i,j)$
end end for $j = 1 : n$
if $j \neq i$
$a(i,j) = a(i,i) \odot a(i,j)$
end end end

Замечание. Алгоритм 2 может рассматриваться как «универсальный алгоритм Флойда-Уоршалла», обобщающий хорошо известные алгоритмы Уоршалла и Флойда для вычисления транзитивного замыкания графа и всех оптимальных путей на графе, см., например, [7]. В свою очередь, эти методы могут рассматриваться как частные случаи алгоритма 2 для тропических и булевых полуколец. Алгоритм 2 также близок к методу «пополнения» Ершова для обращения матриц и решения систем вида $Ax = b$ в классической линейной алгебре, подробности см. [11, глава 2].

2.2. Системы Тёплица

Запишем эскалаторный метод для нахождения решения $x = A^*b$ уравнения $x = Ax \oplus b$, где x и b — вектор-столбцы, аналогичный описанному в предыдущем разделе эскалаторному методу для нахождения A^*. Начнем с $x^{(1)} = A_1^* b_1$. Пусть $x^{(k)}$ – вектор, найденный после $(k-1)$ шага. Представим его в виде

$$x^{(k+1)} = \begin{pmatrix} z \\ x_{k+1} \end{pmatrix}.$$

Используя (2.1), мы получаем

$$x_{k+1} = u_{k+1}(h_k^T x^{(k)} \oplus b_{k+1}), \quad z = x^{(k)} \oplus A_k^* g_k x_{k+1}. \tag{2.1}$$

Требуется вычислить $A_k^* g_k$. Далее мы покажем, что это вычисление может быть проведено гораздо эффективнее, когда A – симметричная тёплицева матрица.

Матрица $A \in \mathrm{Mat}_{nn}(\mathcal{S})$ называется *тёплицевой* (или *матрицей Тёплица*), если существуют скаляры $r_{-n+1}, \ldots, r_0, \ldots, r_{n-1}$, такие, что $A_{ij} = r_{j-i}$ для всех i и j. Другими словами, тёплицевы матрицы обладают тем свойством, что их элементы постоянны вдоль любой линии, параллельной главной диагонали (и вдоль самой главной диагонали). Например, матрица

$$A = \begin{pmatrix} r_0 & r_1 & r_2 & r_3 \\ r_{-1} & r_0 & r_1 & r_2 \\ r_{-2} & r_{-1} & r_0 & r_1 \\ r_{-3} & r_{-2} & r_{-1} & r_0 \end{pmatrix}$$

тёплицева. Такие матрицы не обязательно симметричные, однако они всегда *персимметричные*, то есть симметричные относительно побочной диагонали. Это свойство алгебраически выражается как $A = E_n A^T E_n$, где $E_n = [e_n, \dots, e_1]$. Через e_i мы обозначаем столбец, чей i-й элемент равен $\mathbf{1}$, а другие элементы равны $\mathbf{0}$. Из свойства $E_n^2 = I_n$ (где I_n — единичная матрица $n \times n$) следует, что произведение двух персимметричных матриц является персимметричным. Следовательно, любая степень персимметричной матрицы персимметрична, как и замыкание персимметричной матрицы. Таким образом, если A персимметрична, то

$$E_n A^* = (A^*)^T E_n. \tag{2.2}$$

Далее мы рассматриваем только симметричные тёплицевы матрицы. Рассмотрим уравнение $y = T_n y \oplus r^{(n)}$, где $r^{(n)} = (r_1, \dots, r_n)^T$ и T_n определяется скалярами $r_0, r_1, \dots, r_{n-1}$ так, что $T_{ij} = r_{|j-i|}$ для всех i и j. Это обобщение задачи Юла–Уокера [1]. Предположим, что мы получили наименьшее решение $y^{(k)}$ системы $y = T_k y \oplus r^{(k)}$ для некоторого k такого, что $1 \leqslant k \leqslant n-1$, где T_k — главная $k \times k$ подматрица матрицы T_n. Тогда T_{k+1} можно записать в следующем виде:

$$T_{k+1} = \begin{pmatrix} T_k & E_k r^{(k)} \\ r^{(k)T} E_k & r_0 \end{pmatrix},$$

а $y^{(k+1)}$ и $r^{(k+1)}$ – как

$$y^{(k+1)} = \begin{pmatrix} z \\ \alpha_k \end{pmatrix}, \quad r^{(k+1)} = \begin{pmatrix} r^{(k)} \\ r_{k+1} \end{pmatrix}.$$

Используя (2.1), (2.2) и тождество $T_k^* r^{(k)} = y^{(k)}$, мы получаем, что

$$\begin{aligned} \alpha_k &= (r_0 \oplus r^{(k)T} y^{(k)})^* (r^{(k)T} E_k y^{(k)} \oplus r_{k+1}), \\ z &= E_k y^{(k)} \alpha_k \oplus y^{(k)}. \end{aligned}$$

Обозначим $\beta_k = r_0 \oplus r^{(k)T} y^{(k)}$. Следующая выкладка показывает, что β_k может быть найден рекурсивно, если $(\beta_{k-1}^*)^{-1}$ существует:

$$\begin{aligned} \beta_k &= r_0 \oplus [r^{(k-1)T} \ r_k] \begin{pmatrix} E_{k-1} y^{(k-1)} \alpha_{k-1} \oplus y^{(k-1)} \\ \alpha_{k-1} \end{pmatrix} \\ &= r_0 \oplus r^{(k-1)T} y^{(k-1)} \oplus (r^{(k-1)T} E_{k-1} y^{(k-1)} \oplus r_k) \alpha_{k-1} \\ &= \beta_{k-1} \oplus (\beta_{k-1}^*)^{-1} \odot \alpha_{k-1}^2. \end{aligned} \tag{2.3}$$

Существование $(\beta_{k-1}^*)^{-1}$ не гарантировано, и поэтому мы записываем две версии нашего алгоритма, первая включает (2.3), а вторая не включает. Мы запишем обе эти версии в одной программе и выделим выражения, которые относятся только к первой или только к второй, с помощью комментариев языка MATLAB %1 и %2 соответственно. Собирая выражения для β_k, α_k и z, мы получим следующее рекурсивное

выражение:

$$\begin{aligned} \beta_k &= r_0 \oplus r^{(k)T} y^{(k)}, \qquad \%2 \\ \beta_k &= \beta_{k-1} \oplus (\beta_{k-1}^*)^{-1} \odot \alpha_{k-1}^2, \qquad \%1 \\ \alpha_k &= (\beta_k)^* \odot (r^{(k)T} E_k y^{(k)} \oplus r_{k+1}), \\ y^{(k+1)} &= \begin{pmatrix} E_k y^{(k)} \alpha_k \oplus y^{(k)} \\ \alpha_k \end{pmatrix}. \end{aligned} \tag{2.4}$$

Рекурсия (2.4) — это обобщенная версия метода Дурбина для задачи Юла-Уокера, прототип алгоритма см. в [1, алгоритм 4.7.1].

Алгоритм 3. *Метод Дурбина для задачи Юла–Уокера для уравнений Беллмана с симметричной матрицей Тёплица.*

Вход: r_0 : скаляр,
r : $n-1 \times 1$ вектор;

$y(1) = r_0^* \odot r(1)$
$\beta = r_0 \qquad \%1$
$\alpha = r_0^* \odot r(1)$
for $k = 1 : n-1$
$\beta = r_0 \oplus r(1:k) \odot y(1:k) \qquad \%2$
$\beta = \beta \oplus (\beta^*)^{-1} \odot \alpha^2 \qquad \%1$
$\alpha = \beta^* \odot (r(k:-1:1) \odot y(1:k) \oplus r(k+1))$
$z(1:k) = y(1:k) \oplus y(k:-1:1) \odot \alpha$
$y(1:k) = z(1:k)$
$y(k+1) = \alpha$
end

Выход: вектор y.

В общем случае алгоритм требует $3/2n^2 + O(n)$ операций $\oplus$ и $\odot$ каждой и только $n^2 + O(n)$ операций $\oplus$ и $\odot$, если допускаются инверсии алгебраических замыканий (как обычно, требуется только n таких замыканий в обоих случаях).

Рассмотрим задачу нахождения $x^{(n)} = T_n^* b^{(n)}$, где T_n заданы как выше, а вектор $b^{(n)} = (b_1, \ldots, b_n)$ произволен. Также мы введем вектор-столбец $y^{(k)}$, являющийся решением задачи Юла–Уокера: $y^{(k)} = T_k^* r^{(k)}$. Основная идея состоит в нахождении выражения для $x^{(k+1)} = T_{k+1}^* b^{(k+1)}$, включающих $x^{(k)}$ и $y^{(k)}$. Запишем $x^{(k+1)}$ и $b^{(k+1)}$ следующим образом:

$$x^{(k+1)} = \begin{pmatrix} v \\ \mu_k \end{pmatrix}, \quad b^{(k+1)} = \begin{pmatrix} b^{(k)} \\ b_{k+1} \end{pmatrix}.$$

Используя персимметрию T_k^* и тождества $T_k^* b_k = x^{(k)}$ и $T_k^* r_k = y^{(k)}$, преобразуем выражения (2.1) и получим, что

$$\begin{aligned} \mu_k &= (r_0 \oplus r^{(k)T} y^{(k)})^* \odot (r^{(k)T} E_k x^{(k)} \oplus b_{k+1}), \\ v &= E_k y^{(k)} \mu_k \oplus x^{(k)}. \end{aligned}$$

Коэффициент $r_0 \oplus r^{(k)T} y^{(k)} = \beta_k$ может быть выражен рекурсивно:

$$\beta_k = \beta_{k-1} \oplus (\beta_{k-1}^*)^{-1} \odot (\alpha_{k-1})^2,$$

если замыкание $(\beta_{k-1})^*$ обратимо. Учитывая эту возможность, получаем следующие выражения:

$$\begin{aligned} \beta_k &= r_0 \oplus r^{(k)T} y^{(k)}, \qquad \%2 \\ \beta_k &= \beta_{k-1} \oplus (\beta_{k-1}^*)^{-1} \odot \alpha_{k-1}^2, \qquad \%1 \\ \mu_k &= \beta_k^* \odot (r^{(k)T} E_k x^{(k)} \oplus b_{k+1}), \\ x^{(k+1)} &= \begin{pmatrix} E_k y^{(k)} \mu_k \oplus x^{(k)} \\ \mu_k \end{pmatrix}. \end{aligned} \tag{2.5}$$

Выражения (2.4) и (2.5) дают следующее обобщение алгоритма Левинсона решения линейных симметричных систем Тёплица, см. прототип в [1, алгоритм 4.7.2].

Алгоритм 4. *Метод Левинсона для систем Беллмана с симметричной матрицей Тёплица*

Вход: r_0: скаляр,
r: $1 \times n-1$ вектор-строка;
b: $n \times 1$ вектор-столбец.

$y(1) = r_0^* \odot r(1)$; $\quad x(1) = r_0^* \odot b(1)$;
$\beta = r_0 \qquad \%1$
$\alpha = r_0^* \odot r(1)$
for $k = 1 : n-1$
$\beta = r_0 \oplus r(1:k) \odot y(1:k) \qquad \%2$
$\beta = \beta \oplus (\beta^*)^{-1} \odot \alpha^2 \qquad \%1$
$\mu = \beta^* \odot (r(k:-1:1) \odot x(1:k) \oplus b(k+1))$
$v(1:k) = x(1:k) \oplus y(k:-1:1) \odot \mu$
$x(1:k) = v(1:k)$
$x(k+1) = \mu$
if $k < n-1$
$\alpha = \beta^* \odot (r(k:-1:1) \odot y(1:k) \oplus r(k+1))$
$z(1:k) = y(1:k) \oplus y(k:-1:1) \odot \alpha$
$y(1:k) = z(1:k)$
$y(k+1) = \alpha$
end end
Выход: вектор x.

В общем случае алгоритм требует $(5/2)n^2 + O(n)$ операций $\oplus$ и $\odot$ каждая, и только $2n^2 + O(n)$ операций $\oplus$ и $\odot$, если алгебраические замыкания обратимы (как обычно, требуется только n замыканий в обоих случаях).

2.3. Разложение LDM

Над классическими числовыми полями матричные уравнения вида $AX = B$ можно решать с помощью разложения матрицы A в произведение $A = LDM$, где L и M — нижняя и верхняя треугольные матрицы с единичной диагональю соответственно, а D — диагональная матрица. Мы построим похожее разложение для решения уравнения Беллмана $X = AX \oplus B$ над полукольцами.

В случае системы $AX = B$ над классическими числовыми полями, разложение $A = LDM$ приводит к следующему разложению $AX = B$:

$$LZ = B, \qquad DY = Z, \qquad MX = Y. \tag{2.1}$$

Следовательно,

$$A^{-1} = M^{-1}D^{-1}L^{-1}, \tag{2.2}$$

если матрица A обратима. По существу, достаточно найти матрицы L, D и M, так как после этого линейная система $AX = B$ решается путем комбинации прямой подстановки Z (решение системы уравнений с нижне-треугольной матрицей), тривиальной инверсии диагональной матрицы Y и обратной подстановки X (решение системы уравнений с верхне-треугольной матрицей).

Используя разложение LDM для $AX = B$ как прототип, мы можем записать аналогичное разложение для $X = AX \oplus B$:

$$Z = LZ \oplus B, \qquad Y = DY \oplus Z, \qquad X = MX \oplus Y. \tag{2.3}$$

Тогда

$$A^* = M^*D^*L^*. \tag{2.4}$$

Тройка (L, D, M), состоящая из нижней треугольной, диагональной и верхней треугольной матриц, называется LDM *-разложением* матрицы A, если выполнены соотношения (2.3) и (2.4). Заметим, что в этом случае главные диагонали матриц L и M — нулевые. Другими словами, матрица L является строго нижне-треугольной, а M является строго верхне-треугольной.

Наша универсальная модификация LDM-разложения схожа с универсальной модификацией LU-разложения, предложенной Карре в [18, 19], см. также ниже. Универсальные алгоритмы LDM и LU-разложения можно применять и в случае обычных числовых полей, однако лишь в том случае, если не требуется операция «выбора ведущего элемента» (pivoting). В этом случае они вычисляют классические LDM и LU разложения для обращения матрицы $I - A$.

Если A — симметричная матрица над полукольцом с коммутативным умножением, то количество вычислений может быть уменьшено вдвое, так как M и L переходят друг в друга при транспозиции. Соответствующие алгоритмы будут подробно рассмотрены ниже.

Мы начнем со случая треугольной матрицы $A = L$ (или $A = M$). Тогда нахождение X сводится к прямой (или обратной) замене. В этих случаях уравнение $X = AX \oplus B$ имеет единственное решение, которое может быть найдено с помощью простых алгоритмов, приведенных ниже. В этих алгоритмах B — вектор (обозначенный через b),

однако их (после соответствующей модификации) можно применять и в случае, когда B — матрица любого подходящего размера. Нас интересует случай строго нижнетреугольной матрицы, соответственно строго верхнетреугольной, когда $a_{ij} = 0$ для $i \leqslant j$, соответственно $a_{ij} = 0$ для $i \geqslant j$.

Алгоритм 5. *Прямая подстановка.*

Вход: Строго нижнетреугольная $n \times n$-матрица l;
$n \times 1$ вектор b.

$y = b$
for $k = 2 : n$
$y(k) = l(k, 1 : k-1) \odot y(1 : k-1)$
end

Выход: вектор y.

Алгоритм 6. *Обратная подстановка.*

Вход: Строго верхнетреугольная $n \times n$-матрица m;
$n \times 1$ вектор b.

$y = b$
for $k = n-1 : -1 : 1$
$y(k) = m(k, k+1 : n) \odot y(k+1 : n)$
end

Выход: вектор y.

Оба алгоритма требуют $n^2/2 + O(n)$ операций $\oplus$ и $\odot$ и никаких алгебраических замыканий.

После LDM-разложения нам нужно вычислить замыкание диагональной матрицы. Это делается поэлементно.

Сформулируем алгоритм LDM-разложения, т. е. вычисление матриц L, D и M, удовлетворяющих (2.3) и (2.4). Доказательство будет приведено ниже.

Алгоритм 7. *LDM-разложение (версия 1).*

Вход: $n \times n$-матрица A с элементами $a(i, j)$,
также используется для хранения результата вычислений
и промежуточных результатов вычислений.

for $j = 1 : n-1$
$v(j) = (a(j, j))^*$
$a(j+1 : n, j) = a(j+1 : n, j) \odot v(j)$
$a(j+1 : n, j+1 : n) = a(j+1 : n, j+1 : n) \oplus a(j+1 : n, j) \odot a(j, j+1 : n)$
$a(j, j+1 : n) = v(j) \odot a(j, j+1 : n)$
end

Алгоритм требует $n^3/3 + O(n^2)$ операций $\oplus$ и $\odot$ и $n-1$ операций алгебраического замыкания.

Строго треугольная матрица L записывается в нижнем треугольнике, строго верхнетреугольная матрица M в верхнем треугольнике, и диагональная матрица D на диагонали матрицы, вычисленной алгоритмом 7.

Теперь покажем, что $A^* = M^* D^* L^*$. Заметим, что приведенное ниже доказательство является модификацией доказательства [14], данного для универсального LU-разложения. Идея заключается в том, чтобы определить универсальный аналог «матриц Гаусса» нижне-треугольных или верхне-треугольных, состоящих из одного столбца или одной строки, соответственно).

$$A = \begin{pmatrix} a_{11} & h^{(1)} \\ g^{(1)} & B^{(1)} \end{pmatrix}. \tag{2.5}$$

Можно проверить, что

$$A^* = \begin{pmatrix} \mathbf{1} & h^{(1)} a_{11}^* \\ O_{n-1\times 1} & I_{n-1} \end{pmatrix} \odot$$

$$\begin{pmatrix} a_{11}^* & O_{1\times n-1} \\ O_{n-1\times 1} & (h^{(1)} a_{11}^* g^{(1)} \oplus B^{(1)})^* \end{pmatrix} \begin{pmatrix} \mathbf{1} & O_{1\times n-1} \\ a_{11}^* g^{(1)} & I_{n-1,} \end{pmatrix}, \tag{2.6}$$

где умножение в правой части ведет к выражению, полностью аналогичному (2.1), где $(h^{(1)} a_{11}^* g^{(1)} \oplus B^{(1)})^*$ играет роль u_{k+1}. Здесь и далее $O_{k\times l}$ означает $k \times l$-матрицы, состоящие только из нулей, а I_l означает единичную матрицу размера l. Это также можно переписать как

$$A^* = M_1^* D_1^* (A^{(2)})^* L_1^*, \tag{2.7}$$

$$M_1 = \begin{pmatrix} O & h^{(1)} a_{11}^* \\ O_{(n-1)\times 1} & O_{(n-1)\times(n-1)} \end{pmatrix},$$

$$D_1 = \begin{pmatrix} a_{11} & O_{1\times(n-1)} \\ O_{(n-1)\times 1} & O_{(n-1)\times(n-1)} \end{pmatrix},$$

$$A^{(2)} = \begin{pmatrix} O_{1\times 1} & O_{1\times(n-1)} \\ O_{(n-1)\times 1} & R^{(2)} \end{pmatrix},$$

$$L_1 = \begin{pmatrix} O_{1\times 1} & O_{1\times(n-1)} \\ a_{11}^* g^{(1)} & O_{(n-1)\times(n-1)} \end{pmatrix},$$

$$R^{(2)} = h^{(1)} a_{11}^* g^{(1)} \oplus B^{(1)}. \tag{2.8}$$

Здесь мы используем, в частности, что $L_1^2 = \mathbf{0}$ и $M_1^2 = \mathbf{0}$, и, следовательно, $L_1^* = I \oplus L_1$ и $M_1^* = I \oplus M_1$.

Первый шаг алгоритма 7 ($k = 1$) вычисляет

$$\begin{pmatrix} a_{11} & h^{(1)} a_{11}^* \\ a_{11}^* g^{(1)} & R^{(2)} \end{pmatrix} = A^{(2)} \oplus L_1 \oplus M_1 \oplus D_1, \tag{2.9}$$

что содержит всю нужную информацию.

Теперь мы продолжим с подматрицы $R^{(2)}$ матрицы $A^{(2)}$, раскладывая ее на множители по аналогии с (2.7), и т. д. Опишем формально k-й шаг этой конструкции, отвечающий k-му шагу алгоритма 7. На этом общем шаге мы работаем с

$$A^{(k)} = \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times(n-k+1)} \\ O_{(n-k+1)\times(k-1)} & R^{(k)} \end{pmatrix}, \tag{2.10}$$

где

$$R^{(k)} = h^{(k-1)}(a^{(k-1)}_{k-1,k-1})^* g^{(k-1)} \oplus B^{(k-1)} = \begin{pmatrix} a^{(k)}_{kk} & h^{(k)} \\ g^{(k)} & B^{(k)} \end{pmatrix}. \tag{2.11}$$

Как и на первом шаге, мы представляем

$$(A^{(k)})^* = M_k^* D_k^* (A^{(k+1)})^* L_k^*, \tag{2.12}$$

где

$$\begin{aligned} M_k &= \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times 1} & O_{(k-1)\times(n-k)} \\ O_{1\times(k-1)} & O_{1\times 1} & h^{(k)}(a^{(k)}_{kk})^* \\ O_{(n-k)\times(k-1)} & O_{(n-k)\times 1} & O_{(n-k)\times(n-k)} \end{pmatrix}, \\ D_k &= \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times 1} & O_{(k-1)\times(n-k)} \\ O_{1\times(k-1)} & a^{(k)}_{kk} & O_{1\times(n-k)} \\ O_{(n-k)\times(k-1)} & O_{(n-k)\times 1} & O_{(n-k)\times(n-k)} \end{pmatrix}, \\ L_k &= \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times 1} & O_{(k-1)\times(n-k)} \\ O_{1\times(k-1)} & O_{1\times 1} & O_{1\times(n-k)} \\ O_{(n-k)\times(k-1)} & (a^{(k)}_{kk})^* g^{(k)} & O_{(n-k)\times(n-k)} \end{pmatrix}, \\ A^{(k+1)} &= \begin{pmatrix} O_{k\times k} & O_{k\times(n-k)} \\ O_{(n-k)\times k} & R^{(k+1)} \end{pmatrix}, \\ R^{(k+1)} &= h^{(k)}(a^{(k)}_{kk})^* g^{(k)} \oplus B^{(k)}. \end{aligned} \tag{2.13}$$

Заметим, что имеется следующее рекуррентное соотношение на элементы $A^{(k)}$:

$$a^{(k+1)}_{ij} = \begin{cases} \mathbf{0}, & \text{если } i \leqslant k \text{ или } j \leqslant k, \\ a^{(k)}_{ij} \oplus a^{(k)}_{ik}(a^{(k)}_{kk})^* a^{(k)}_{kj}, & \text{если } i > k \text{ и } j > k. \end{cases} \tag{2.14}$$

Эта рекурсия сразу видна в алгоритме 7. Более того, можно показать по индукции, что матрица, вычисленная на k-м шаге алгоритма, равна

$$A^{(k+1)} \oplus \bigoplus_{i=1}^{k} L_i \oplus \bigoplus_{i=1}^{k} M_i \oplus \bigoplus_{i=1}^{k} D_i. \tag{2.15}$$

Иначе говоря, эта матрица составлена из $h^{(1)}a_{11}^*$, ..., $h^{(k)}(a^{(k)}_{kk})^*$ (в верхнем треугольнике), $a_{11}^* g^{(1)}$, ..., $(a^{(k)}_{kk})^* g^{(k)}$ (в нижнем треугольнике), $a_{11}, \ldots, a^{(k)}_{kk}$ (на диагонали) и $R^{(k+1)}$ (в юго-восточном углу).

После раскрытия всех выражений (2.12) для $A^{(k)}$, где $k = 1, \ldots, n$, мы получаем

$$A^* = M_1^* D_1^* \cdots M_n^* D_n^* L_n^* \cdots L_1^* \tag{2.16}$$

(на самом деле, $M_n = L_n = \mathbf{0}$ и, следовательно, $M_n^* = L_n^* = I$). Заметив, что D_i^* и M_j^* коммутируют при $i < j$, мы можем переписать

$$A^* = M_1^* \cdots M_n^* D_1^* \cdots D_n^* L_n^* \cdots L_1^*. \tag{2.17}$$

Рассмотрим равенства

$$\begin{aligned} (D_1 \oplus \ldots \oplus D_n)^* &= D_1^* \cdots D_n^*, \\ (L_1 \oplus \ldots \oplus L_n)^* &= L_n^* \cdots L_1^*, \\ (M_1 \oplus \ldots \oplus M_n)^* &= M_1^* \cdots M_n^*. \end{aligned} \tag{2.18}$$

Первое из этих равенств очевидно. По поводу остальных двух заметим, что $M_k^2 = L_k^2 = \mathbf{0}$ для всех k, следовательно, $M_k^* = I \oplus M_k$ и $L_k^* = I \oplus L_k$. Далее, $L_i L_j = \mathbf{0}$ при $i < j$ и $M_i M_j = 0$ при $i > j$. Используя эти равенства, можно показать, что

$$\begin{aligned} (L_1 \oplus \ldots \oplus L_n)^* &= \bigoplus_{i=0}^{n-1} (L_1 \oplus \ldots \oplus L_n)^i = \\ &= (I \oplus L_n) \cdots (I \oplus L_1) = L_n^* \cdots L_1^*, \\ (M_1 \oplus \ldots \oplus M_n)^* &= \bigoplus_{i=0}^{n-1} (M_1 \oplus \ldots \oplus M_n)^i = \\ &= (I \oplus M_1) \cdots (I \oplus M_n) = M_1^* \cdots M_n^*, \end{aligned} \tag{2.19}$$

что дает два последних равенства из (2.18). Заметим, что в (2.19) мы использовали нильпотентность $L_1 \oplus \ldots \oplus L_n$ и $M_1 \oplus \ldots \oplus M_n$, что позволяет применить (1.6).

Можно проверить, что матрицы $M := M_1 \oplus \ldots \oplus M_n$, $L := L_1 \oplus \ldots \oplus L_n$ и $D := D_1 \oplus \ldots \oplus D_n$ содержатся в верхнем треугольнике, в нижнем треугольнике и, соответственно, на диагонали матрицы, вычисленной алгоритмом 7. Эти матрицы удовлетворяют LDM-разложению $A^* = M^* D^* L^*$. Этим завершается объяснение алгоритма 7.

В терминах матричных вычислений алгоритм 7 есть версия LDM-разложения с внешним произведением. Этот алгоритм может быть переделан в форму, почти идентичную с [1, алгоритм 4.1.1]:

Алгоритм 8. *LDM-разложение (версия 2).*

Вход: $n \times n$-матрица A с элементами $a(i, j)$,
также используется для хранения результата вычислений
и промежуточных результатов вычислений.

for $j = 1 : n$
$v(1 : j) = a(1 : j, j)$
for $k = 1 : j - 1$

$v(k+1:j) = v(k+1:j) \oplus a(k+1:j,k) \odot v(k)$
end
for $i = 1 : j-1$
$a(i,j) = (a(i,i))^* \odot v(i)$
end
$a(j,j) = v(j)$
for $k = 1 : j-1$
$a(j+1:n,j) = a(j+1:n,j) \oplus a(j+1:n,k) \odot v(k)$
end
$d = (v(j))^*$
$a(j+1:n,j) = a(j+1:n,j) \odot d$
end

Этот алгоритм выполняет в точности те же операции, что и алгоритм 7, вычисляя последовательно один за другим столбцы результата. А именно, в первой части основного цикла он вычисляет элементы $a_{ij}^{(i)}$ для $i = 1, \dots, j$, сначала под видом элементов v и окончательно в присваивании «$a(i,j) = (a(i,i))^* \odot v(i)$». Сложность этого алгоритма такая же, как у алгоритма 7.

2.4. LDM-разложение с симметрией и ленточной структурой

Когда матрица A симметрична, т. е. $a_{ij} = a_{ji}$ для всех i, j, естественно ожидать, что LDM-разложение тоже должно быть симметрично, т. е. $M = L^T$. Действительно, повторяя рассуждения предыдущего раздела, можно показать, что все промежуточные матрицы $A^{(k)}$ симметричны, следовательно, $M_k = L_k^T$ для всех k и $M = L^T$. Теперь мы представим две версии симметричного LDM-разложения, отвечающие двум версиям LDM-разложения, данным в предыдущем разделе. Заметим, что объем вычислений в этих алгоритмах примерно вдвое меньше, чем в их полных версиях. В обоих случаях они требуют $n^3/6 + O(n^2)$ операций $\oplus$ и $\odot$ (каждый) и $n-1$ операций взятия алгебраического замыкания.

Алгоритм 9. *Симметричное LDM-разложение (версия 1).*

Вход: $n \times n$-матрица A с элементами $a(i,j)$,
также используется для хранения результата вычислений
и промежуточных результатов вычислений.

for $j = 1 : n-1$
$v(j) = (a(j,j))^*$
for $k = j+1 : n$
for $l = j+1 : k$
$a(k,l) = a(k,l) \oplus a(k,j) \odot v(j) \odot a(l,j)$
end
end
$a(j+1:n,j) = a(j+1:n,j) \odot v(j)$
end

Строго треугольная матрица L содержит нижний треугольник результата, а матрица D — на диагонали. Следующая версия обобщает [1, алгоритм 4.1.2].

Алгоритм 10. *Симметричное LDM-разложение (версия 2).*

Вход: $n \times n$-матрица A с элементами $a(i,j)$,
также используется для хранения результата вычислений
и промежуточных результатов вычислений.

for $j = 1 : n$
for $i = 1 : j-1$
$v(i) = (a(i,i)^*)^{-1} a(i,j)$
end
$v(j) = v(j) \oplus a(j, 1:j-1) \odot v(1:j-1)$
$a(j,j) = v(j)$
for $k = 1 : j-1$
$a(j+1:n, j) = a(j+1:n, j) \oplus a(j+1:n, k) \odot v(k)$
end
$d = (v(j))^*$
$a(j+1:n, j) = a(j+1:n, j) \odot d$ **end**

Заметим, что эта версия требует обратимости замыканий $a(i,i)^*$, вычисленных алгоритмом.

Замечание. В случае идемпотентного полукольца мы имеем $(D^*)^2 = D^*$, следовательно, $A^* = (M^*D^*)(D^*L^*)$. Когда A симметрична, мы можем записать $A^* = (G^*)^T G^*$, где $G = D^*L$. Очевидно, эта **идемпотентное разложение Холецкого** приводит лишь к небольшой модификации алгоритма 9 (или 10). См. также классическое разложение Холецкого в [1, алгоритм 4.2.2].

$A = (a_{ij})$ называется **ленточной матрицей** с верхней шириной полосы q и нижней шириной полосы p, если $a_{ij} = 0$ для всех $j > i+p$ и всех $i > j+p$. Ленточная матрица с $p = q = 1$ называется **тридиагональной**. Для обоснования универсальной версии LDM-разложения ленточных матриц нам нужно показать, что параметры лент матриц $A = A^{(2)}, \ldots, A^{(k)}$, вычисленных в процессе LDM-разложения, не больше параметров матрицы $A^{(1)} = A$. Положим по индукции, что $A = A^{(1)}, \ldots, A^{(k)}$ имеют требуемые параметры ленты и рассмотрим элементы $a_{ij}^{(k+1)}$ для $i > j+p$. Если $i \leqslant k$ или $j \leqslant k$, то $a_{ij}^{(k+1)} = \mathbf{0}$, поэтому мы можем предположить, что $i > k$ и $j > k$. В этом случае $i > k+p$, следовательно, $a_{ik}^{(k)} = \mathbf{0}$ и

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} \oplus a_{ik}^{(k)} (a_{kk}^{(k)})^* a_{kj}^{(k)} = \mathbf{0}.$$

Таким образом, мы показали, что нижняя полоса $A^{(k)}$ не больше p. Таким же образом можно показать, что ее верхняя полоса не превышает q. Мы используем это для построения ленточной версии LDM-разложения, её прототип см. в [1, алгоритм 4.3.1].

Алгоритм 11. *LDM-разложение ленточной матрицы.*

A — ленточная $n \times n$-матрица с элементами $a(i,j)$, нижняя ширина ленты p и верхняя ширина ленты q также используется для хранения окончательного результата и промежуточных результатов вычислений.

for $j = 1 : n-1$
$v(j) = (a(j,j))^*$
for $i = j+1 : \min(j+p, n)$
$a(i,j) = a(i,j) \odot v(j)$
end
for $k = j+1 : \min(j+q, n)$
for $i = j+1 : \min(j+p, n)$
$a(k,j) = a(k,j) \oplus a(k,i) \odot a(i,j)$
end end
for $k = j+1 : \min(j+q, n)$
$a(j,k) = v(j) \odot a(j,k)$
end end

Когда p и q фиксированы и $n >> p, q$ переменная, легко видеть, что алгоритм выполняет примерно npq каждой из операций $\odot$ и $\oplus$.

Замечание. *Есть некоторые еще более специальные виды ленточных матриц, например, матрицы Хессенберга и трёхдиагональные матрицы. Матрицы Хессенберга определяются как ленточные матрицы с $p = 1$ и $q = n$, а в случае трёхдиагональных матриц $p = q = 1$. Несложно записать дальнейшие адаптации алгоритма 11 для этих случаев.*

2.5. Итерационные схемы

По настоящему универсальную итерационную схему трудно найти, так как условия, при которых такие схемы работают, а также условия остановки для таких схем зависят от полукольца и от представления данных. Тем не менее, основные операции таких схем бывают универсальны и очень просты.

Рассмотрим следующий итерационный процесс решения уравнения Беллмана $X = AX \oplus B$:

$$X^{(k+1)} = AX^{(k)} \oplus B. \tag{2.1}$$

Итерируя выражения (2.1) для всех k от 0 до $m-1$, мы получаем

$$X^{(m)} = A^m X^{(0)} \oplus \bigoplus_{i=0}^{m-1} A^i B. \tag{2.2}$$

Таким образом, результат зависит от поведения $A^m X^{(0)}$. Алгоритм может быть переписан следующим образом (для случая, когда B — вектор-столбец).

Алгоритм 12. *Итерации Якоби*

Вход: $n \times n$-матрица A с элементами $a(i,j)$;
$n \times 1$ вектор-столбцы b и x

situation = 'вычисление'
while situation == 'вычисление'
$x = A \odot x \oplus b$
situation = *newsituation*(...)
if situation == 'расходимость'
disp('Итерации Якоби расходятся.')
exit
end
if situation == 'сходимость'
disp('Решение найдено:')
exit
end end
Выход: ситуация, x.

Далее мы кратко обсудим поведение итераций Якоби над обычной арифметикой с неотрицательными вещественными числами и над полукольцом $\mathbf{R}_{\max}$. Для простоты в обоих случаях мы ограничимся случаем *неприводимой* матрицы A, то есть когда ассоциированный граф сильно связан.

Хорошо известно, что над обычной арифметикой (в неприводимом неотрицательном случае) итерации Якоби сходятся тогда и только тогда, когда наибольшее собственное значение $r(A)$ матрицы A строго меньше 1. Это связано, в частности, с тем, как ведет себя последовательность $\{A^m x^{(0)}\}_{m\geqslant 0}$. Заметим, что итерации Якоби, даже в случае сходимости, как правило, дают лишь приближённое решение уравнения $x = Ax + b$.

В случае $\mathbf{R}_{\max}$ поведение последовательности $\{A^m x^{(0)}\}_{m\geqslant 0}$ отличается от случая обычной неотрицательной алгебры. Однако, как и в случае неотрицательной алгебры, это поведение зависит от $r(A)$, наибольшего собственного значения A в алгебре макс-плюс (то есть относительно собственной задачи $A \odot x = \lambda \odot x$ в этой алгебре). А именно $A^m x^{(0)} \to \mathbf{0}$ и, следовательно, итерации сходятся если $r(A) < \mathbf{1}$. Более того, в этом случае $A^* = (I \oplus A \oplus \ldots \oplus A^{n-1})$ и, следовательно, итерации доставляют **точное** решение уравнения Беллмана после **конечного** числа шагов. Наоборот, $A^m x^{(0)} \to +\infty$ и, следовательно, итерации расходятся при $r(A) > 1$. Подробности см., например, [18].

На границе $r(A) = \mathbf{1}$ степени A^m достигают периодического режима после конечного числа шагов. Следовательно, последовательность $\{A^* b \oplus A^m x^{(0)}\}_{m\geqslant 0}$ также становится периодической после конечного числа шагов. Если период равен единице, то есть если эта последовательность стабилизируется, то метод сходится к некоторому общему решению уравнения $x = Ax \oplus b$, которое описывается как суперпозиция A^*b и некоторого собственного вектора матрицы A [2,17]. Вектор A^*b может доминировать, в этом случае метод сходится к A^*b как «ожидалось». Однако, период последовательности (после выхода на периодический режим) может быть и больше единицы, в этом случае итерации Якоби не доставляют никакого решения уравнения $x = Ax \oplus b$ непосредственно, однако такое решение может быть найдено с помощью суммирования по периоду. Степени матриц и теория собственных векторов и собственных значений над алгеброй макс-плюс подробно описаны в монографии Бутковича [16], см. также [49,50].

В более сложной схеме итераций Гаусса–Зейделя мы также можем использовать предварительно найденные координаты $X^{(k)}$. В этом случае матрица A записывается в виде $L \oplus U$, где L — строго нижнетреугольная часть A, а U — верхнетреугольная часть с диагональю. Итерации записываются как

$$X^{(k)} = LX^{(k)} \oplus UX^{(k-1)} \oplus B = L^*UX^{(k-1)} \oplus L^*B. \tag{2.3}$$

Заметим, что преобразования правой части однозначно, так как L — строго нижнетреугольная матрица, и L^* однозначно определяется как $I \oplus L \oplus \ldots \oplus L^{n-1}$ (где n — размерность A). Другими словами, мы только применяем формальную подстановку. Итерируя выражения (2.3) для всех k вплоть до m, мы получим

$$X^{(m)} = (L^*U)^m X^{(0)} \oplus \bigoplus_{i=0}^{m-1} (L^*U)^i L^*B. \tag{2.4}$$

Правая часть напоминает формулу $(L \oplus U)^* = (L^*U)^*L^*$, см. (1.1), поэтому естественно ожидать, что эти итерации сходятся к A^*B при хорошем выборе $X^{(0)}$. Результат зависит от поведения $(L^*U)^m X^{(0)}$. Алгоритм может быть записан следующим образом (мы снова полагаем, что B — вектор-столбец).

Алгоритм 13. *Итерации Гаусса-Зейделя*

Вход: $n \times n$-матрица A с элементами $a(i,j)$;
$n \times 1$-вектор-столбцы b и x

situation = 'вычисление'
while situation == 'вычисление'
for $i = 1 : n$
$y(i) = a(i, i : n) \odot x(i : n) \oplus b(i)$
end
for $i = 2 : n$
$x(i) = a(i, 1 : i-1) \odot x(1 : i-1)$
‘ **end**
situation = *newsituation*(...)
if situation == 'расходимость'
disp('Итерации Гаусса-Зейделя расходятся.')
exit
end
if situation == 'сходимость'
disp('Решение найдено:')
exit
end end

Выход: ситуация, x.

Анализ работы схемы Гаусса-Зейделя в случаях тропической (макс-плюс) алгебры и неотрицательной линейной алгебры мы здесь не приводим, однако ясно, что его можно провести по аналогии с итерациями Якоби.

2.6. LU-разложение

Обсудим более кратко универсальную версию разложения $A = LU$, которая была описана в работах Б. А. Карре [18]. Этот раздел содержит лишь формулировки соответствующих алгоритмов, так как доказательства были даны в работах Карре и Бекхауса [14, 18, 19] и могут быть получены (как и сами алгоритмы) путем небольшой модификации соответствующих рассуждений в разделе про LDM-разложение (см. выше).

Напомним, что в случае классической системы $AX = B$ над числовыми полями разложение $A = LU$, где L — нижняя унитреугольная (с диагональными элементами равными 1), а U — верхняя треугольная матрица, приводит к следующему разложению исходного уравнения:

$$LZ = B, \quad UX = Z.$$

Таким образом, мы получаем

$$A^{-1} = U^{-1}L^{-1}$$

при условии, что матрица A обратима. Достаточно найти L и U, так как после этого линейная система (2.6.) может быть решена комбинацией прямой подстановки для Z и обратной подстановки для X.

В случае системы $X = AX \oplus B$ над полукольцами, следуя [18], мы можем записать

$$Z = LZ \oplus B, \quad X = UX \oplus Z.$$

Таким образом,

$$A^* = U^*L^*,$$

где U — верхне-треугольная, а L — строго нижне-треугольная матрица.

Далее мы приводим алгоритмы прямой и обратной подстановки, а также две версии универсального LU-разложения (то есть одновременного вычисления матриц L и U).

Алгоритм 14. *Прямая подстановка.*

Вход: Строго нижнетреугольная $n \times n$-матрица l;
$n \times 1$ вектор b.

$y = b$
for $k = 2 : n$
$y(k) = l(k, 1 : k-1) \odot y(1 : k-1)$
end

Выход: вектор y.

Алгоритм 15. *Обратная подстановка.*

Вход: Верхнетреугольная $n \times n$-матрица u;
$n \times 1$ вектор b.

$y = b$
$y(n) = (u(n, n))^* \odot y(n)$
for $k = n-1 : -1 : 1$

$y(k) = u(k, k+1:n) \odot y(k+1:n)$
$y(k) = (u(k,k))^* \odot y(k)$
end

Выход: вектор y.

Отметим, что обратная подстановка отличается от случая LDM-разложения, так как матрица U (в отличие от M) является лишь треугольной, а не строго треугольной. Прямые подстановки для обеих схем, разумеется, идентичны.

Алгоритм 16. *LU-разложение (версия 1).*

Вход: $n \times n$-матрица A с элементами $a(i,j)$, также используется для хранения результата вычислений и промежуточных результатов вычислений.

for $j = 1 : n-1$
$v(j) = (a(j,j))^*$
$a(j+1:n, j) = a(j+1:n, j) \odot v(j)$
$a(j+1:n, j+1:n) = a(j+1:n, j+1:n) \oplus a(j+1:n, j) \odot a(j, j+1:n)$
end

Алгоритм 17. *LU-разложение (версия 2).*

Вход: $n \times n$-матрица A с элементами $a(i,j)$, также используется для хранения результата вычислений и промежуточных результатов вычислений.

for $j = 1 : n$
$v(1:j) = a(1:j, j)$
for $k = 1 : j-1$
$v(k+1:j) = v(k+1:j) \oplus a(k+1:j, k) \odot v(k)$
end
$a(1:j, j) = v(1:j)$
for $k = 1 : j-1$
$a(j+1:n, j) = a(j+1:n, j) \oplus a(j+1:n, k) \odot v(k)$
end
$d = (v(j))^*$
$a(j+1:n, j) = a(j+1:n, j) \odot d$
end

Эти алгоритмы основаны на следующих определениях, аналогичных (2.12) и (2.13). Полагая

$$A^{(1)} := A = \begin{pmatrix} a_{11} & h^{(1)} \\ g^{(1)} & B^{(1)} \end{pmatrix},$$

для любого $k = 1, \dots, n$ получаем

$$(A^{(k)})^* = U_k^* (A^{(k+1)})^* L_k^*, \tag{2.1}$$

где

$$U_k = \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times 1} & O_{(k-1)\times(n-k)} \\ O_{1\times(k-1)} & a_{kk}^{(k)} & h^{(k)} \\ O_{(n-k)\times(k-1)} & O_{(n-k)\times 1} & O_{(n-k)\times(n-k)} \end{pmatrix},$$

$$L_k = \begin{pmatrix} O_{(k-1)\times(k-1)} & O_{(k-1)\times 1} & O_{(k-1)\times(n-k)} \\ O_{1\times(k-1)} & O_{1\times 1} & O_{1\times(n-k)} \\ O_{(n-k)\times(k-1)} & (a_{kk}^{(k)})^* g^{(k)} & O_{(n-k)\times(n-k)} \end{pmatrix}, \tag{2.2}$$

$$A^{(k+1)} = \begin{pmatrix} O_{k\times k} & O_{k\times(n-k)} \\ O_{(n-k)\times k} & R^{(k+1)} \end{pmatrix},$$

$$R^{(k+1)} = h^{(k)}(a_{kk}^{(k)})^* g^{(k)} \oplus B^{(k)} = \begin{pmatrix} a_{k+1,k+1}^{(k+1)} & h^{(k+1)} \\ g^{(k+1)} & B^{(k+1)} \end{pmatrix}.$$

2.7. Программная реализация универсальных алгоритмов

Программные реализации универсальных алгоритмов на полукольцах не могут быть такими же эффективными, как и стандартные программные реализации (относительно скорости вычислений), но они намного более гибкие. Программные модули могут иметь дело с абстрактными (и переменными) операциями и типами данных. Конкретные значения этих операций и типов данных могут быть заданы соответствующими входными данными. В этом случае конкретные операции и типы данных порождаются дополнительными программными модулями. В программах, написанных в таком духе, удобно использовать специальные методы так называемого объектно-ориентированного (и функционального) дизайна, см., например, [41,47,52]. К счастью, недавно появились мощные инструменты, поддерживающие объектно-ориентированный дизайн программ, включающие компиляторы для реальных и удобных языков программирования (например, C++ и Java) и современные системы компьютерной алгебры. С недавних пор, этот тип программных методов был назван обобщенным программированием (см., например, [47]).

Реализация на C++. Используя шаблоны и объектно-ориентированное программирование, А. В. Чуркин и С. Н. Сергеев [9] создали программу (на Visual C++), которая демонстрирует, как некоторые универсальные алгоритмы вычисляют замыкания матриц A^* и решают уравнения Беллмана $x = Ax \oplus b$ в различных полукольцах. Программа также может решать обычную систему $Ax = b$ в обычной арифметике, преобразовывая ее в форму «Беллмана». Перед нажатием кнопки «Solve» пользователь должен выбрать полукольцо, задачу и алгоритм. После этого начальные данные записываются в матрице (для удобства отображения размерность матрицы ограничена 10×10). Результат может быть и вектором, и матрицей в зависимости от решаемой задачи. Объектно-ориентированный подход позволяет реализовать различные полукольца как объекты с различными определениями основных операций, тогда как сама программа алгоритма записана компактно и в единственном «экземпляре» (шаблоне).

Примеры полуколец. Полукольцо соответствует объекту, используемому универсальным алгоритмом, и определяет конкретную реализацию этого алгоритма. Пользователь может выбрать одно из следующих полуколец:

1) $\oplus = +$ и $\otimes = \times$: обычная арифметика над действительными числами;
2) $\oplus = \max$ и $\otimes = +$: арифметика макс-плюс над $\mathbb{R} \cup \{-\infty\}$;
3) $\oplus = \min$ и $\otimes = +$: арифметика мин-плюс над $\mathbb{R} \cup \{+\infty\}$;
4) $\oplus = \max$ и $\otimes = \times$: арифметика макс-умножить над неотрицательными числами;
5) $\oplus = \max$ и $\otimes = \min$: арифметика макс-мин над вещественным отрезком $[a, b]$ (концы a и b может выбирать пользователь);
6) $\oplus = \mathrm{OR}$ и $\otimes = \mathrm{AND}$: булева логика над двухэлементном множеством $\{0, 1\}$.

Алгоритмы. Пользователь может выбрать один из следующих методов:

1) **Метод исключения Гаусса**, включающий универсальные реализации эскалаторного метода (алгоритм 1), Флойда-Уоршалла (алгоритм 2, алгоритм Ершова (основанный на прототипе из [11, гл. 2]) и универсальный алгоритм Ротэ [48];

2) **Методы для систем Тёплица**, включающие универсальные реализации схем Дурбина и Левинсона (алгоритмы 3 и 4);

3) LDM**-разложение** (алгоритм 7) и его адаптации к симметрическим (алгоритм 9), ленточным (алгоритм 11), трёхдиагональным матрицам и матрицам Хессенберга.

4) **Итерационные схемы** Якоби и Гаусса-Зейделя. Как было упомянуто выше, эти схемы не универсальны, т. к. критерии остановки и условия, при которых они сходятся и дают правильный ответ, различны для обычной арифметики и идемпотентных полуколец (в частности, макс-плюс).

Типы матриц. Пользователь может выбрать работу с общими матрицами или с матрицами специального вида, например, симметричные, симметричные тёплицевы, ленточные, Хессенберга и тридиагональные.

Отображение. В случае идемпотентных полуколец матрица может быть отображена как взвешенный направленный граф. После выполнения вычислений пользователь может захотеть найти оптимальный путь между данной парой вершин или отобразить дерево оптимальных путей. Эти задачи могут быть решены, используя родительские ссылки, как в случае с классическим методом Флойда-Уоршалла, вычисляющим все оптимальные пути. См., например, [7]. В нашем случае механизм родительских ссылок может быть реализован прямо в классе, описывающем идемпотентную арифметику.

Другие арифметики и интервальные расширения. Возможна также реализация различные типов арифметики как типов данных и их сочетание с выбором полуколец. Более того, все реализованные полукольца могут быть расширены до их интервальных версий. Такие возможности не были реализованы в программе А. В. Чуркина и С. Н. Сергеева [9], они отложены до следующей версии. Список этих арифметик включает целые числа, дробную арифметику с цепными дробями и управляемой точностью.

Реализации в MATLAB. Вся работа (кроме инструментов отображения) была повторена в MATLAB [9], который также обладает возможностями объектно-ориентированного программирования. Разумеется, универсальные алгоритмы, записанные в MATLAB, очень близки к описанным в настоящей статье.

Перспективы на будущее. По-видимому, основная задача заключается в том, чтобы, используя идемпотентный принцип соответствия и вычисления над полукольцами, разработать программную систему широкого назначения, основанную на коллекции универсальных алгоритмов. Разработка такой системы на основе универсальных алгоритмов может гарантировать сокращение рабочего времени для программистов и пользователей благодаря унификации программного обеспечения. Также может быть гарантирована произвольная точность и безопасность численных вычислений. При этом надо учесть, что высокоуровневые инструменты, такие как STL [47, 52], обладают и очевидными преимуществами, и некоторыми недостатками, поэтому должны использоваться с осторожностью.

Система должна содержать несколько уровней (включая уровни пользователя и программиста) и большое количество модулей. Грубо говоря, она должна быть разделена на три части. Первая часть должна содержать модули, которые реализуют модули доменов (конечные представления основных математических объектов). Вторая часть должна реализовывать универсальные (инвариантные) методы вычисления. Третья часть должна содержать модули, реализующие алгоритмы, которые зависят от модели. Эти модули могут быть использованы в пользовательских программах на C++, Java, Maple, MATLAB и др.

В частности, система должна содержать следующие модули:

— Доменные модули:

  – целые числа произвольной длины;
  – рациональные числа;
  – рациональные числа конечной точности (см. [8]);
  – комплексные рациональные числа конечной точности;
  – рациональные числа с фиксированной и плавающей запятой;
  – комплексные рациональные числа;
  – вещественные числа с плавающей запятой произвольной точности;
  – комплексные числа произвольной точности;
  – p-адические числа;
  – интервальные числа;
  – кольца многочленов над различными кольцами;
  – идемпотентные полукольца;
  – интервальные идемпотентные полукольца;
  – и другие.

— Алгоритмы:

  – линейная алгебра;
  – численное интегрирование;
  – корни многочленов;
  – интерполяция сплайнами и аппроксимации;
  – рациональные и полиномиальные интерполяции и аппроксимации;
  – вычисление специальных функций;
  – дифференциальные уравнения;
  – оптимизации и оптимальное управление;

– идемпотентный функциональный анализ;
– и другие.

Эта программная система может быть особенно полезна разработчикам алгоритмов, программистам, студентам и математикам.

References

[1] Дж. Голуб, Ч. Ван Лоун, *Матричные вычисления*, Мир, М., 2000; англ. пер.:G. H. Golub and C. van Loan, *Matrix Computations*, John Hopkins Univ. Press, London, 1989.

[2] Н. К. Кривулин, *Методы идемпотентной алгебры в задачах моделирования и анализа сложных систем*, Изд-во СПбГУ, Санкт-Петербург, 2009. [N. K. Krivulin, *Methods of Idempotent Algebra in Problems of Modeling and Analysis of Complex Systems*, Publishing House of St. Petersburg university, St. Petersburg, 2009 (In Russian)].

[3] Г. Л. Литвинов, А. Н. Соболевский, "Точные интервальные решения дискретного уравнения Беллмана и полиномиальная сложность задач идемпотентной линейной алгебры", *Доклады РАН*, **374**:3 (2000), 304–306, arXiv: org/abs/math.LA/0101041. [G. L. Litvinov, A. N. Sobolevsky, "Exact interval solutions of the discrete Bellman equation and polynomial complexity of idempotent linear algebra problems", *Doklady Mathematics*, **374**:3 (2000), 304–306 (In Russian), arXiv: org/abs/math.LA/0101041].

[4] В. П. Маслов, "Новый подход к обобщённым решениям нелинейных систем", *Доклады АН СССР*, **292**:1 (1987), 29–33. [V. P. Maslov, "A new approach to generalized solutions of nonlinear systems", *Doklady Mathematics*, **292**:1 (1987), 29–33 (In Russian)].

[5] В. П. Маслов, "О новом принципе суперпозиции для задач оптимизации", *УМН*, **42**:3(255) (1987), 39–48; англ. пер.:V. P. Maslov, "On a new principle of superposition for optimization problems", *Russian Mathematical Surveys*, **42**:3 (1987), 43–54.

[6] В. П. Маслов, В. Н. Колокольцов, *Идемпотентный анализ и его применение в оптимальном управлении*, Наука, М., 1994; англ. пер.:V. P. Maslov, V. N. Kolokoltsov, *Idempotent Analysis and Its Applications*, Springer Science+Business Media Dordrecht, Dordrecht, 1997.

[7] Р. Седжвик, *Алгоритмы на C++*, часть 5: *Алгоритмы на графах*, Диасофт, Киев, 2002; англ. пер:R. Sedgewick, *Algorithms in C++*, part 5: *Graph Algorithms*, 3rd, Addison-Wesley Publishing Company, Inc., California, New York, Amsterdam, 2002.

[8] S. Sergeev, "Universal algorithms for generalized discrete matrix Bellman equations with symmetric Toeplitz matrix", *Tambov University Reports. Series: Natural and Technical Sciences*, **16**:6–2 (2011), 1751–1758, arXiv: org/abs/math/0612309.

[9] С. Н. Сергеев, А. В. Чуркин, "Программа для демонстрации универсальных алгоритмов решения дискретного уравнения Беллмана в различных полукольцах", *Idempotent and tropical mathematics and problems of mathematical physics. II*, Международный семинар «Idempotent and Tropical Mathematics and Problems of Mathematical Physics» (Независимый московский университет, российско-французская лаборатория «J.-V. Poncelet», Россия, 25–30 августа), Независимый московский университет, М., 2007, 107–109, arXiv: org/abs/0709.4119. [S. N. Sergeev, A. V. Chourkin, "Universal algorithms solving discrete Bellman systems of equations over semirings", *Idempotent and Tropical Mathematics and Problems of Mathematical Physics. II*, International Workshop "Idempotent and Tropical Mathematics and Problems of Mathematical Physics" (Independent University of Moscow, French–Russian Laboratory "J.-V. Poncelet", Russia, August 25–30), Independent University of Moscow, Moscow, 2007, 107–109 (In Russian), arXiv: org/abs/0709.4119].

[10] А. Н. Соболевский, "Интервальная арифметика и линейная алгебра над идемпотентными полукольцами", *Доклады РАН*, **369**:6 (1999), 747–749. [A. N. Sobolevsky, "Interval arithmetic and linear algebra over idempotent semirings", *Doklady Mathematics*, **369**:6 (1999), 747–749 (In Russian)].

[11] Д. К. Фаддеев, В. Н. Фаддеева, *Вычислительные методы линейной алгебры*, 3-е изд., Изд-во «Лань», Санкт-Петербург, 2002. [D. K. Faddeev, V. N. Faddeeva, *Computational Methods of Linear Algebra*, 3rd ed., Publishing house "Doe", St. Petersburg, 2002 (In Russian)].

[12] G. Alefeld and J. Herzberger, *Introduction to interval computations*, Academic Press, New York, 1983.

[13] F. L. Baccelli, G. Cohen, G. J. Olsder and J. P. Quadrat, *Synchronization and Linearity: an Algebra for Discrete Event Systems*, Wiley and Sons, 1992.

[14] R. C. Backhouse, B. A. Carré, "Regular algebra applied to path-finding problems", *Journal of the Institute of Mathematics and its Applications*, **15** (1975), 161–186.

[15] W. Barth, E. Nuding, "Optimale Lösung von Intervalgleichungsystemen", *Computing*, **12** (1974), 117–125.

[16] P. Butkovič, *Max-linear Systems: Theory and Algorithms*, Springer, London, 2010.

[17] P. Butkovič, H. Schneider and S. Sergeev, "Z-matrix equations in max algebra, nonnegative linear algebra and other semirings", 2011, arXiv: org/abs/1110.4564.

[18] B. A. Carré, "An algebra for network routing problems", *Journal of the Institute of Mathematics and its Applications*, **7** (1971), 273–294.

[19] B. A. Carré, *Graphs and Networks*, Oxford Univ. Press, Oxford, 1979.

[20] K. Cechlárová and R. A. Cuninghame-Green, "Interval systems of max-separable linear equations", *Linear Algebra and its Applications*, **340**:1–3 (2002), 215–224.

[21] R. A. Cuninghame-Green, *Minimax Algebra*, Lecture Notes in Economics and Mathematical Systems, **166**, Springer, Berlin, 1979.

[22] M. Fiedler, J. Nedoma, J. Ramík, J. Rohn and K. Zimmermann, *Linear optimization problems with inexact data*, Springer, New York, 2006.

[23] J. Golan, *Semirings and Their Applications*, Kluwer Academic Publishers, 2000.

[24] M. Gondran, "Path algebra and algorithms", *Combinatorial Programming: Methods and Applications*, Proceedings of the NATO Advanced Study Institute held at the Palais des Congres (Versailles, France, 2–13 September, 1974), Reidel, Dordrecht, 1975, 137–148.

[25] M. Gondran and M. Minoux, *Graphs et Algorithms*, Éditions Eylrolles, Paris, 1979.

[26] M. Gondran and M. Minoux, *Graphs, Dioids and Semirings*, Springer, New York, 2010.

[27] J. Gunawardena, *Idempotency*, Cambridge Univ. Press, Cambridge, 1998.

[28] L. Hardouin, B. Cottenceau, M. Lhommeau, and E. Le Corronc, "Interval systems over idempotent semiring", *Linear Algebra and its Applications*, **431** (2009), 855–862.

[29] V. Kreinovich, A. Lakeev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Applied Optimization, Kluwer Academic Publishers, Dordrecht, 1998.

[30] D. J. Lehmann, "Algebraic structures for transitive closure", *Theoretical Computer Science*, **4** (1977), 59–76.

[31] G. L. Litvinov, "The Maslov dequantization, idempotent and tropical mathematics: a brief introduction", *Journal of Mathematical Sciences*, **141**:4 (2007), 1417–1428, arXiv: org/abs/math.GM/0507014.

[32] G. L. Litvinov and V. P. Maslov, "1210–1211", *Russian Mathematical Surveys*, **51** (1996).

[33] G. L. Litvinov and V. P. Maslov, "The correspondence principle for idempotent calculus and some computer applications", *Idempotency*, Cambridge Univ. Press, Cambridge, 1998, 420–443, arXiv: org/abs/math.GM/0101021.

[34] G. L. Litvinov and V. P. Maslov, *Idempotent mathematics and mathematical physics*, **307**, AMS Contemporary Mathematics, Providence, 2005.

[35] G. L. Litvinov, V. P. Maslov, A. Ya. Rodionov, and A. N. Sobolevskiĭ, *Universal algorithms, mathematics of semirings and parallel computations*, Lecture Notes in Computational Science and Engineering, **75**, 2011, arXiv: org/abs/1005.1252.

[36] G. L. Litvinov and E. V. Maslova, "Universal numerical algorithms and their software implementation", *Programming and Computer Software*, **26**:5 (2000), 275–280, arXiv: org/abs/math.SC/0102114.

[37] G. L. Litvinov, A. Ya. Rodionov and A. V. Tchourkin, "Approximate rational arithmetic and arbitrary precision computations for universal algorithms", *International Journal of Pure and Applied Mathematics*, **45**:2 (2008), 193–204, arXiv: org/abs/math.NA/0101152.

[38] G. L. Litvinov and S. N. Sergeev, *Tropical and Idempotent Mathematics*, **495**, AMS Contemporary Mathematics, Providence, 2009.

[39] G. L. Litvinov and A. N. Sobolevskiĭ, "Idempotent interval analysis and optimization problems", *Reliable Computing*, **7**:5 (2001), 353–377, arXiv: org/abs/math.SC/0101080.

[40] G. L. Litvinov, V. P. Maslov and A. Ya. Rodionov, "A unifying approach to software and hardware design for scientific calculations and idempotent mathematics", 2000, arXiv: org/abs/math.SC/0101069.

[41] M. Lorenz, *Object oriented software: a practical guide*, Prentice Hall Books, Englewood Cliffs, New Jersey, 1993.

[42] G. Mikhalkin, "Tropical geometry and its applications", *Proceedings of the Madrid ICM*. V. 2, 2006, 827–852, arXiv: org/abs/math.AG/0601041.

[43] R. E. Moore, *Methods and applications of interval analysis*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, 1979.

[44] H. Myškova, "Interval systems of max-separable linear equations", *Linear Algebra and its Applications*, **403** (2005), 263–272.

[45] H. Myškova, "Control solvability of interval systems of max-separable linear equations", *Linear Algebra And Its Applications*, **416**:2–3 (2006), 215–223.

[46] A. Neumaier, *Interval methods for systems of equations*, Cambridge University Press, Cambridge, 1990.

[47] I. Pohl, *Object-Oriented Programming Using C++*, 2nd ed., Addison-Wesley, Reading, 1997.

[48] G. Rote, "A systolic array algorithm for the algebraic path problem", *Computing*, **34** (1985), 191–219.

[49] S. Sergeev, "Max-algebraic attraction cones of nonnegative irreducible matrices", *Linear Algebra And Its Applications*, **435**:7 (2011), 1736–1757, arXiv: org/abs/math.AG/0903.3960.

[50] S. Sergeev and H. Schneider, "CSR expansions of matrix powers in max algebra", *Transactions of Amer. Math. Soc.*, **364** (2012), 5969–5994, arXiv: org/abs/math.AG/0912.2534.

[51] I. Simon, "Recognizable sets with multiplicities in the tropical semiring", *Lecture Notes in Computer Science*, **324** (1988), 107–120.

[52] A. Stepanov and M. Lee, *The Standard Template Library*, Hewlett-Packard Company, Palo Alto, 1994.

[53] O. Viro, "Dequantization of real algebraic geometry on logarithmic paper", *European Congress of Mathematics*, European Congress of Mathematics (Barcelona, 2000, July 10–14), Progress in Mathematics, Birkhäuser, Basel, 135–146, arXiv: org/abs/math/0005163.

[54] O. Viro, "From the sixteenth Hilbert problem to tropical geometry", *Japanese Journal of Mathematics*, **3**:2 (2008), 185–214.

[55] V. V. Voevodin, *Mathematical Foundations of Parallel Computings*, World Scientific Publ. Co., Singapore, 1992.

Информация об авторах

Литвинов Григорий Лазаревич, доктор физико-математических наук, профессор. Институт проблем передачи информации им. А. А. Харкевича Российской академии наук, г. Москва, Российская Федерация. E-mail: glitvinov@gmail.com

Information about the authors

Grigory L. Litvinov, Doctor of Physics and Mathematics, Professor. Institute for Information Transmission Problems of the Russian Academy of Sciences, Moscow, the Russian Federation. E-mail: glitvinov@gmail.com

Родионов Анатолий Яковлевич, преподаватель. Московский центр непрерывного математического образования, г. Москва, Российская Федерация. E-mail: ayarodionov@yahoo.com

Сергеев Сергей, доцент Школы математики. Университет Бирмингема, г. Бирмингем, Великобритания.
E-mail: sergeevs@maths.bham.ac.uk

Соболевский Андрей Николаевич, доктор физико-математических наук, профессор РАН, директор Института проблем передачи информации имени А. А. Харкевича РАН. Институт проблем передачи информации имени А. А. Харкевича РАН, г. Москва, Российская Федерация. E-mail: sobolevski@iitp.ru
ORCID: http://orcid.org/0000-0002-3082-5113

Конфликт интересов отсутствует.

Для контактов:
Литвинов Григорий Лазаревич
E-mail: glitvinov@gmail.com

Поступила в редакцию 14 августа 2019 г.
Поступила после рецензирования 24 октября 2019 г.
Принята к публикации 29 ноября 2019 г.

Anatoliy Ya. Rodionov, Teacher. Moscow Center for Continuous Mathematical Education, Moscow, the Russian Federation. E-mail: ayarodionov@yahoo.com

Sergeĭ Sergeev, Associate Professor of the Mathematics School. University of Birmingham, Birmingham, United Kingdom.
E-mail: sergeevs@maths.bham.ac.uk

Andrei N. Sobolevsky, Doctor of Physics and Mathematics, Professor of RAS, Director of the Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute). Institute for Information Transmission Problems of the Russian Academy of Sciences, Moscow, the Russian Federation. E-mail: sobolevski@iitp.ru
ORCID: http://orcid.org/0000-0002-3082-5113

There is no conflict of interests.

Corresponding author:
Grigory L. Litvinov
E-mail: glitvinov@gmail.com

Received 14 August 2019
Reviewed 24 October 2019
Accepted for press 29 November 2019