ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2017, том 57, № 12, с. 1933–1945

УДК 519.61

ОБРАТНЫЕ ЛИНЕЙНЫЕ РАЗНОСТНЫЕ ОПЕРАТОРЫ[1)]

© 2017 г. С. А. Абрамов

(119333 Москва, ул. Вавилова, 40, ВЦ ФИЦ ИУ РАН)

e-mail: sergeyabramov@mail.ru

Поступила в редакцию 28.08.2016 г.
Переработанный вариант 23.01.2017 г.

Для матриц, элементами которых служат скалярные линейные разностные операторы, предлагаются алгоритмы проверки обратимости (унимодулярности) и построения обратной матрицы в случае ее существования. Эти алгоритмы имеют меньшую сложность в сравнении с известными алгоритмами. Рассматриваются отличия предлагаемых алгоритмов от их дифференциальных аналогов. Библ. 33.

Ключевые слова: сложность алгоритмов, разностный оператор, операторная матрица, унимодулярная матрица, проверка унимодулярности, построение обратной матрицы.

DOI: 10.7868/S004446691712002X

1. ВВЕДЕНИЕ

Для матриц над некоторым полем или кольцом задачи проверки обратимости и фактического построения обратной матрицы, когда она существует, могут быть отнесены к классическим математическим задачам. Ниже эти задачи рассматриваются применительно к операторным матрицам. В данном случае элементами матриц служат скалярные линейные разностные операторы с коэффициентами из некоторого разностного поля $\mathbb{K}$ с автоморфизмом (сдвигом) σ. Предполагается, что поле $\mathbb{K}$ имеет характеристику 0. Обсуждаются новые алгоритмы решения названных задач. Следует оговориться, что эти задачи могут решаться известными алгоритмами, предназначенными изначально для решения более общих задач, и об этом еще будет сказано ниже. Обсуждаемые новые алгоритмы имеют меньшую сложность.

Обычно в случаях операторных матриц вместо “обратимая матрица” употребляется термин *унимодулярная* матрица. Мы также будем пользоваться этим термином.

Алгоритмы проверки унимодулярности и построения обратной матрицы для дифференциального случая, когда $\mathbb{K}$ является дифференциальным полем характеристики 0 с дифференцированием $\partial = '$ и когда элементы матриц суть скалярные линейные дифференциальные операторы над $\mathbb{K}$, рассматривались автором в [1]. Для данной операторной матрицы L как дифференциальные, так и обсуждаемые ниже разностные алгоритмы основаны на рассмотрении размерности пространства решений V_L соответствующей системы уравнений в предположении, что компоненты решений принадлежат связанному с L расширению Пикара-Вессио (см. [2]–[4]) поля $\mathbb{K}$. Операторная матрица L полного ранга (строки L независимы над кольцом скалярных линейных операторов) является унимодулярной, если и только если $\dim V_L = 0$, т.е. само пространство V_L является нулевым (см. [5]).

Будем в дальнейшем придерживаться следующих обозначений. Кольцо $n \times n$-матриц (n – большее нуля целое) с элементами в некотором кольце или поле R обозначается через $\mathrm{Mat}_n(R)$. Если M – некоторая $n \times n$-матрица, то обозначение $M_{i,*}$ при $1 \le i \le n$ используется для $1 \times n$-матрицы, равной i-й строке матрицы M. Диагональная $n \times n$-матрица с элементами $r_1, \ldots, r_n$ на диагонали обозначается через $\mathrm{diag}(r_1, \ldots, r_n)$, единичная $n \times n$-матрица – через I_n.

Результаты статьи предварительно анонсировались в расширенной аннотации [6].

[1)]Работа выполнена при финансовой поддержке РФФИ (код проекта 16-01-001174).

2. ОТЛИЧИЕ ОТ ДИФФЕРЕНЦИАЛЬНОГО СЛУЧАЯ

Имеется, по меньшей мере, три отличия от рассмотренного в [1] дифференциального случая.

1. В разностном случае возникает естественная возможность раздельного рассмотрения сложностей по числу арифметических операций и по числу сдвигов, подобно алгоритмам сортировки, для которых раздельно рассматривают сложности по числу сравнений и по числу перемещений элементов. Подсчитывая для данной матрицы число требуемых арифметических операций, мы можем игнорировать сдвиги, и соответственно, занимаясь числом сдвигов – игнорировать арифметические операции. В дифференциальном случае это раздельное рассмотрение двух сложностей не столь просто, так как применение оператора дифференцирования ∂ слева к скалярному дифференциальному оператору требует дополнительных арифметических операций (предполагаем, что для операторов используется стандартное представление в виде полиномов от оператора дифференцирования ∂): например, $\partial(a\partial + b) = a\partial^2 + (a' + b)\partial + b'$. Игнорирование операций дифференцирования может привести к неправильным оценкам числа арифметических операций. В разностном же случае мы имеем дело с автоморфизмом σ, что открывает возможность раздельного рассмотрения сложностей.

2. Простая замена ∂ на σ в алгоритмах для дифференциального случая не приводит к алгоритмам для разностного случая, здесь требуются дополнительные ухищрения. Уже говорилось, что основой алгоритмов является определение значения $\dim V_L$. Ситуация такова, что система $\sigma y = Ay$, где A – это $n \times n$-матрица с элементами из $\mathbb{K}$, имеет пространство решений размерности n если и только если матрица A невырожденна, а в дифференциальном случае $y' = Ay$ эта невырожденность не требуется. Однако это вычисление размерности и в разностном случае может быть проведено для произвольной линейной разностной системы алгоритмически (см. [5]).

3. Пусть разностное поле $\mathbb{K}$ является полем рациональных функций от x и применение σ к произвольной рациональной функции получается подстановкой $x + 1$ вместо x с последующим приведением результата к канонической форме (для σ^{-1} подставляется $x - 1$). Тогда естественно считать, что σ имеет ту же сложность, что и σ^k при любом целом k. Принятие этой точки зрения может повлиять на сложность алгоритма по числу сдвигов. В дифференциальном случае отождествление сложностей для ∂ и ∂^k было бы безосновательным: естественно считать, что применение ∂^k – это k-кратное применение ∂.

3. АДЕКВАТНЫЕ РАЗНОСТНЫЕ РАСШИРЕНИЯ

Напомним, что *разностное кольцо* – это коммутативное кольцо $\mathbb{K}$ с единицей и с автоморфизмом σ (который мы часто будем называть *сдвигом*). Если при этом $\mathbb{K}$ является полем, то оно называется соответственно *разностным полем*. Рассматриваемые в дальнейшем разностные поля без оговорок предполагаются полями характеристики 0.

Кольцом констант разностного кольца $\mathbb{K}$ называется $\mathrm{Const}(\mathbb{K}) = \{c \in \mathbb{K} \mid \sigma c = c\}$. В случае когда $\mathbb{K}$ является разностным полем, $\mathrm{Const}(\mathbb{K})$ будет подполем поля $\mathbb{K}$ (*полем констант* поля $\mathbb{K}$).

Пусть $\mathbb{K}$ – разностное поле с автоморфизмом σ, кольцо Λ – разностное расширение поля $\mathbb{K}$ (на $\mathbb{K}$ соответствующий автоморфизм кольца Λ совпадает с σ, для этого автоморфизма кольца Λ используется то же самое обозначение σ).

Определение 1. Будем говорить, что кольцо Λ, являющееся разностным расширением поля $\mathbb{K}$, представляет собой *адекватное* разностное расширение поля $\mathbb{K}$, если $\mathrm{Const}(\Lambda)$ является полем и для произвольной системы

$$\sigma y = Ay, \quad y = (y_1, \ldots, y_n)^{\mathrm{T}} \tag{1}$$

с невырожденной матрицей $A \in \mathrm{Mat}_n(\mathbb{K})$ размерность линейного над полем $\mathrm{Const}(\Lambda)$ пространства принадлежащих Λ^n решений равна n.

Невырожденность матрицы A в этом определении существенна: если, например, A имеет нулевую первую строку, то любое решение системы (1) будет иметь нулевую компоненту y_1.

Замечание 1. Так называемый q-разностный случай (см. [7], [8]) охватывается общим разностным случаем.

Если поле $\mathrm{Const}(\mathbb{K})$ алгебраически замкнуто, то существует с точностью до разностного изоморфизма (т.е. изоморфизма, перестановочного с σ) единственное адекватное расширение Ω такое, что $\mathrm{Const}(\Omega) = \mathrm{Const}(\mathbb{K})$. Такое расширение называется *универсальным* разностным кольцевым расширением (Пикара–Вессио) поля $\mathbb{K}$. Полное доказательство его существования непросто, см. [4, разд. 1.4]. Существование же некоторого адекватного разностного расширения Λ для произвольного разностного поля доказывается достаточно элементарно, см. [9, разд. 5.1]. Но, подчеркнем, для адекватного расширения равенство $\mathrm{Const}(\Lambda) = \mathrm{Const}(\mathbb{K})$ не гарантируется, в общем случае $\mathrm{Const}(\mathbb{K})$ является для $\mathrm{Const}(\Lambda)$ собственным подполем.

Утверждение о существовании универсального разностного расширения для произвольного разностного поля характеристики 0 становится неверным, если понимать расширение как поле. Известный пример Ч. Франке (см. [10]) – скалярное уравнение $\sigma y = -y$ над полем $\mathbb{K}$ с алгебраически замкнутым полем констант. Это уравнение не имеет ненулевых решений в любом разностном расширении, имеющем алгебраически замкнутое поле констант.

Далее мы будем обозначать через Λ фиксированное адекватное разностное расширение разностного поля $\mathbb{K}$ с автоморфизмом σ.

4. ПОРЯДКИ ОПЕРАТОРОВ, ПРОСТРАНСТВА РЕШЕНИЙ

Скалярный разностный оператор является элементом кольца $\mathbb{K}[\sigma, \sigma^{-1}]$. Для ненулевого скалярного оператора $f = \sum a_i \sigma^i$ мы определяем его верхний и нижний порядки:

$$\overline{\mathrm{ord}} f = \max\{i | a_i \neq 0\}, \quad \underline{\mathrm{ord}} f = \min\{i | a_i \neq 0\}$$

и порядок

$$\mathrm{ord}\, f = \overline{\mathrm{ord}} f - \underline{\mathrm{ord}} f.$$

Полагаем $\overline{\mathrm{ord}}\, 0 = -\infty$, $\underline{\mathrm{ord}}\, 0 = \infty$, $\mathrm{ord}\, 0 = -\infty$.

Для конечного множества F скалярных операторов (вектора, матрицы, строки матрицы) мы определим $\overline{\mathrm{ord}}\, F$ как максимум верхних порядков его элементов, $\underline{\mathrm{ord}} F$ – как минимум нижних порядков его элементов и, наконец, $\mathrm{ord}\, F$ – как $\overline{\mathrm{ord}}\, F - \underline{\mathrm{ord}}\, F$.

Разностная операторная матрица – это матрица, принадлежащая $\mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$. В ходе последующего изложения мы будем сопоставлять такой операторной матрице некоторые матрицы, принадлежащие $\mathrm{Mat}_n(\mathbb{K})$. Чтобы избежать терминологической путаницы, мы будем коротко называть операторные матрицы просто *операторами*. Случай скалярных операторов будем оговаривать особо.

Оператор имеет полный ранг (или является оператором полного ранга), если его строки линейно независимы над $\mathbb{K}[\sigma, \sigma^{-1}]$. Поясним, что строки $u_1, \ldots, u_s$ одинаковой длины, элементы которых принадлежат $\mathbb{K}[\sigma, \sigma^{-1}]$, называются *линейно зависимыми* (более подробно: линейно зависимыми над $\mathbb{K}[\sigma, \sigma^{-1}]$), если существуют такие одновременно не равные нулю $f_1, \ldots, f_s \in \mathbb{K}[\sigma, \sigma^{-1}]$, что $f_1 u_1 + \ldots + f_s u_s = 0$; в противном случае эти строки называются *линейно независимыми* (над $\mathbb{K}[\sigma, \sigma^{-1}]$).

Если

$$L \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}]), \quad l = \overline{\mathrm{ord}}\, L, \quad t = \underline{\mathrm{ord}}\, L,$$

и L – ненулевая матрица, то L можно записать в развернутом виде:

$$L = A_l \sigma^l + A_{l-1} \sigma^{l-1} + \cdots + A_t \sigma^t, \tag{2}$$

где $A_l, A_{l-1}, \ldots, A_t \in \mathrm{Mat}_n(\mathbb{K})$, при этом матрицы A_l, A_t (ведущая и трейлинговая матрицы исходного оператора) ненулевые.

Определение 2. Пусть верхние порядки строк оператора L равны $\alpha_1, \ldots, \alpha_n$, а нижние равны $\beta_1, \ldots, \beta_n$. *Фронтальной* матрицей оператора L назовем ведущую матрицу оператора PL, где

$$P = \operatorname{diag}(\sigma^{l-\alpha_1}, \ldots, \sigma^{l-\alpha_n}), \quad l = \overline{\operatorname{ord}}\, L.$$

Соответственно *тыльной* матрицей оператора L назовем трейлинговую матрицу оператора QL, где

$$Q = \operatorname{diag}(\sigma^{t-\beta_1}, \ldots, \sigma^{t-\beta_n}), \quad t = \underline{\operatorname{ord}}\, L.$$

Будем говорить, что оператор L строго редуцирован, если невырожденны как фронтальная, так и тыльная матрицы этого оператора.

Определение 3. Оператор $L \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ называется *унимодулярным* или *обратимым*, если для него существует обратный $L^{-1} \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$: $LL^{-1} = L^{-1}L = I_n$. Множество унимодулярных $n \times n$-операторов будет обозначаться через Υ_n. Два оператора $L_1, L_2 \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ назовем *эквивалентными*, если $L_1 = UL_2$ для некоторого $U \in \Upsilon_n$.

Обозначение V_L будет привлекаться для пространства принадлежащих Λ^n (см. разд. 3) решений системы $L(y) = 0$. Для краткости будем иногда говорить о V_L как о пространстве решений оператора L.

Переформулируем для разностного случая теорему 1 из [1]:

Теорема 1. *Пусть оператор $L \in \operatorname{Mat}_n(K[\sigma, \sigma^{-1}])$ имеет полный ранг. Тогда верно следующее*:

(i) *если оператор L строго редуцирован, то* $\dim V_L = \sum_{i=1}^{n} \operatorname{ord} L_{i,*}$.;

(ii) $L \in \Upsilon_n \Leftrightarrow V_L = 0$.

Как и в дифференциальном случае, доказательство основывается на публикациях [5], [9].

5. РЕДУКЦИЯ ОПЕРАТОРОВ

В этом разделе и далее без специальных оговорок будет предполагаться, что n и d – целые числа, $n > 0$, $d \geqslant 0$, при этом исходный оператор L таков, что

$$L \in \operatorname{Mat}_n(K[\sigma, \sigma^{-1}]), \quad l = \overline{\operatorname{ord}}\, L, \quad t = \underline{\operatorname{ord}}\, L, \quad d = \operatorname{ord} L. \tag{3}$$

Алгоритмам решения рассматриваемых матричных задач сопоставляются две сложности, являющиеся функциями n и d:

• *арифметическая сложность*, т.е. число арифметических операций в поле $\mathbb{K}$ в худшем случае при фиксированных n, d;

• *сдвиговая сложность*, т.е. число применений операций σ, σ^{-1} к элементам поля $\mathbb{K}$ в худшем случае при фиксированных n, d.

При выводе асимптотических оценок сложности, наряду с O-нотацией мы будем прибегать к Θ-нотации (см. [11]): соотношение $f(n,d) = \Theta(g(n,d))$ равносильно тому, что $f(n,d) = O(g(n,d))$ и одновременно с этим $g(n,d) = O(f(n,d))$, т.е. $f(n,d)$ и $g(n,d)$ суть величины одного порядка при $n, d \to \infty$.

В связи с понятием сдвиговой сложности уточним, что вычисление $\sigma^k a$ для $k \in \mathbb{Z}$, $a \in \mathbb{K}$ по предположению требует $|k|$ применений операции σ или σ^{-1}.

5.1. EG-алгоритмы

5.1.1. Алгоритм EG_σ. Алгоритм EG_σ (см. [12]–[14]) строит для L полного ранга такой эквивалентный оператор $\hat{L}_+ \in \operatorname{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$, что $\overline{\operatorname{ord}}\, \hat{L}_+ = \overline{\operatorname{ord}}\, L$, $\underline{\operatorname{ord}}\, \hat{L}_+ \geqslant \underline{\operatorname{ord}}\, L$, оператор $\hat{L}_+$ имеет невырожденную ведущую матрицу. Если ранг L не полон, то алгоритм сообщает об этом.

Алгоритм EG_σ строит $\widehat{L}_+$ на месте L, т.е. шаг за шагом L изменяется, постепенно превращаясь в $\widehat{L}_+$.

Проверить, являются ли строки ведущей матрицы оператора L линейно независимыми над $\mathbb{K}$. Если да, то L не изменяется и алгоритм останавливается. Иначе выполняется серия однотипных шагов, на каждом из которых производятся следующие действия:

• вычислить коэффициенты $p_1, \ldots, p_n \in \mathbb{K}$ зависимости строк ведущей матрицы;

• среди строк оператора, которым соответствуют ненулевые коэффициенты, выбрать имеющую наименьший нижний порядок (если таких строк несколько, то взять какую-нибудь одну из них); пусть i – номер выбранной строки;

• заменить строку $L_{i,*}$ на $\sum_{k=1}^{n} p_k L_{k,*}$. Если исходный оператор L имеет полный ранг, то получается ненулевая i-я строка. Пусть верхний порядок получившейся i-й строки равен α; применить $\sigma^{l-\alpha}$ к этой строке.

Коль скоро исходный оператор имеет полный ранг, мы получим $\widehat{L}_+$ с невырожденной ведущей матрицей после не более, чем nd шагов; если ранг L не полон, то не позднее, чем на шаге с номером nd в операторе возникнет нулевая строка. В самом деле, сумма нижних порядков всех строк оператора увеличивается с каждым шагом. Но она изначально не меньше, чем tn, и на любом из шагов не может стать больше, чем ln. Имеем $ln - tn = dn$. Если выполнено $nd - 1$ шагов и при этом не возникло нулевой строки, то это означает, что на шаге с номером nd сумма нижних порядков всех строк оператора станет равной ln, и сам оператор будет матрицей из $\mathrm{Mat}_n(\mathbb{K})$; этот шаг будет последним: здесь либо полученная матрица невырожденна, либо ее строки линейно зависимы и тогда на этом же шаге мы приходим к матрице, имеющей нулевую строку.

Замечание 2. Если тыльная матрица оператора L невырожденна, то она останется невырожденной и после применения EG_σ. Это обеспечивается выбором для исключения тех строк, которые имеют наименьшие нижние порядки. Заметим дополнительно, что если трейлинговая матрица оператора L невырожденна, то одновременно эта матрица является тыльной невырожденной матрицей для L.

То, что оператор $\widehat{L}_+$ эквивалентен оператору L и, как следствие, $V_L = V_{\widehat{L}_+}$, выводится из существования автоморфизма σ^{-1}, обратного для σ.

5.1.2. Алгоритм $\mathrm{EG}_{\sigma^{-1}}$. По аналогии с EG_σ можно предложить алгоритм $\mathrm{EG}_{\sigma^{-1}}$, в котором вместо ведущей матрицы рассматривается трейлинговая матрица оператора, линейной комбинацией строк заменяется строка с наибольшим верхним порядком и к полученной линейной комбинации строк применяется σ в соответствующей отрицательной степени. Если L имеет полный ранг, то после не более, чем nd шагов получим эквивалентный оператор $\widehat{L}_-$ с невырожденной трейлинговой матрицей; при этом $\underline{\mathrm{ord}}\, \widehat{L}_- = \underline{\mathrm{ord}}\, L$, $\overline{\mathrm{ord}}\, \widehat{L}_- \leqslant \overline{\mathrm{ord}}\, L$. Если ранг L не полон, то после не более, чем nd шагов алгоритма в операторе возникнет нулевая строка. В случае невырожденной фронтальной матрицы оператора L мы получаем $\widehat{L}_-$ также с невырожденной фронтальной матрицей, – благодаря выбору для исключения строк с наибольшими верхними порядками. По аналогии с замечанием 2 заметим, что если ведущая матрица оператора L невырожденна, то одновременно эта матрица является для L фронтальной (невырожденной) матрицей.

Замечание 3. Таким образом, если L имеет полный ранг, то последовательное применение $\mathrm{EG}_{\sigma^{-1}}$ и EG_σ позволяет получить для исходного оператора эквивалентный ему оператор в строго редуцированной форме. В силу теоремы 1(i) мы можем вычислить $\dim V_L$.

5.2. Сложность EG-алгоритмов

Предложение 1. *Арифметическая сложность для каждого из алгоритмов* EG_σ, $\mathrm{EG}_{\sigma^{-1}}$ *допускает оценку*

$$\Theta(n^{\omega+1}d + n^3d^2), \tag{4}$$

где ω *– показатель матричного умножения,* $2 < \omega \leqslant 3$. *В то же время для сдвиговой сложности имеет место оценка*

$$\Theta(n^2d^2). \tag{5}$$

Доказательство. Поиск линейной зависимости строк ведущей или трейлинговой матрицы сводится к поиску решения однородной системы n линейных алгебраических уравнений с n неизвестными. Сложность этого поиска есть $\Theta(n^\omega)$. Для понижения порядка строки на единицу потребуется в худшем случае $\Theta(n^\omega + n^2d)$ арифметических операций, это дает оценку (4). Пусть для исходного оператора L порядки всех строк равны d. Если каждый шаг алгоритма уменьшает порядок одной из строк на единицу и по окончании работы алгоритма мы получаем $\widehat{L}$ порядка 0 (это соответствует худшему случаю), то на каждую из строк будет затрачено $nd + n(d-1) + \ldots + n = (nd^2)$ сдвигов, а на все строки – число сдвигов, допускающее оценку (5). Предложение доказано.

Применение к оператору L алгоритма $\mathrm{EG}_{\sigma^{-1}}$ и, в случае полного ранга, последующего применения алгоритма EG_σ позволяет найти $\dim V_L$, см. замечание 3. По теореме 1(ii) оператор L унимодулярен, если и только если $\dim V_L = 0$, и проверка унимодулярности может быть выполнена с арифметической сложностью (4) и сдвиговой сложностью (5). В разд. 6.1 мы покажем, что арифметическую сложность этой проверки можно уменьшить.

5.3. RR-алгоритмы

5.3.1. Алгоритмы RR_σ и $\mathrm{RR}_{\sigma^{-1}}$. Для оператора L полного ранга алгоритм RR_σ (см. [16]–[18]) строит такой эквивалентный оператор $\breve{L}_+ \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$, что $\overline{\mathrm{ord}}\,\breve{L}_+ \leqslant \overline{\mathrm{ord}}\,L$, $\underline{\mathrm{ord}}\,\breve{L}_+ \geqslant \underline{\mathrm{ord}}\,L$, который имеет невырожденную фронтальную матрицу. Если ранг оператора L не полон, то алгоритм сообщает об этом.

Алгоритм RR_σ строит $\breve{L}_+$ на месте L, т.е. L изменяется шаг за шагом, постепенно превращаясь в $\breve{L}_+$.

Проверить, являются ли строки фронтальной матрицы для L линейно независимыми над $\mathbb{K}$. Если да, то L не изменяется и алгоритм останавливается. Иначе выполняется серия однотипных шагов, на каждом из которых производятся следующие действия:

а) вычислить коэффициенты $p_1, \ldots, p_n \in \mathbb{K}$ зависимости строк фронтальной матрицы;

б) среди строк, которым соответствуют ненулевые коэффициенты зависимости, выбрать имеющую наибольший верхний порядок (если таких строк несколько, то взять какую-нибудь одну из них). Пусть i – номер выбранной строки;

в) заменить строку $L_{i,*}$ на $\sum_{k=1}^{n} (\sigma^{\alpha_i - l} p_k)(\sigma^{\alpha_i - \alpha_k} L_{k,*})$.

Если L имеет полный ранг, то после не более, чем nd шагов получим $\breve{L}_+$ с невырожденной фронтальной матрицей; если ранг L не полон, то после не более, чем nd шагов в операторе возникнет нулевая строка.

По аналогии с $\mathrm{EG}_{\sigma^{-1}}$, можно предложить алгоритм $\mathrm{RR}_{\sigma^{-1}}$, который строит эквивалентный оператор $\breve{L}_-$ с невырожденной тыльной матрицей.

5.3.2. Расширенные RR_σ и $\mathrm{RR}_{\sigma^{-1}}$. Расширенный алгоритм RR_σ позволяет, в частности, наряду с $\breve{L}_+$ найти такой унимодулярный оператор $U \in \Upsilon_n$, что $\breve{L}_+ = UL$.

Фактически, этот алгоритм, который мы назовем ExtRR_σ, кроме L, получает на вход произвольный оператор $W \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ и в том случае, когда L имеет полный ранг, находит $\breve{L}_+$ и UW, где U – упомянутый унимодулярный оператор (если $W = I_n$, то вместе с $\breve{L}_+$ получаем U):

Применить RR_σ к L и попутно продублировать над оператором, изначально равным W, все выполняемые над L операции.

Предложение 2. *Пусть* $L, W \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$, *оператор* L *имеет полный ранг,* $\operatorname{ord} L = d$, $\operatorname{ord} W = d_1$. *Тогда* $\operatorname{ord} W = O(nd + d_1)$ *на каждом шаге алгоритма* ExtRR_σ.

Доказательство. Утверждение является следствием [16, предложение 1], [22, теорема 4.9], различие между дифференциальным и разностным случаями здесь несущественно.

Все сказанное сохраняет силу для алгоритма $\mathrm{RR}_{\sigma^{-1}}$, расширенного аналогичным образом. Этот расширенный вариант мы назовем $\mathrm{ExtRR}_{\sigma^{-1}}$.

Замечание 4. В [1, предложение 5] для дифференциального случая доказано, что для унимодулярного $n \times n$-оператора L порядка d выполнено неравенство

$$\operatorname{ord} L^{-1} \leqslant (n-1)d \tag{6}$$

и что при любых n, d существует оператор L, для которого $\operatorname{ord} L^{-1} = (n-1)d$. Доказательство легко модифицируется для разностного случая.

Пример 1. При $n = 2$ для любого унимодулярного оператора L выполнено $\operatorname{ord} L = \operatorname{ord} L^{-1}$, например:

$$\begin{pmatrix} 1 & -\dfrac{1}{x}\sigma \\ \dfrac{x^2}{2} & -\dfrac{x}{2}\sigma + 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 - \dfrac{(x+1)^2}{2x}\sigma & \dfrac{1}{x}\sigma \\ -\dfrac{x^2}{2} & 1 \end{pmatrix}. \tag{7}$$

В данном случае

$$\mathbb{K} = \mathbb{Q}(x), \quad \sigma x = x + 1, \quad \operatorname{ord} L = \operatorname{ord} L^{-1} = 1.$$

5.3.3. Сложность RR-алгоритмов.

Предложение 3. *Арифметическая сложность каждого из алгоритмов* RR_σ *и* $RR_{\sigma^{-1}}$ *допускает оценку*

$$\Theta(n^{\omega+1}d + n^3d^2). \tag{8}$$

Для сдвиговой сложности имеет место оценка

$$\Theta(n^3d^3). \tag{9}$$

Доказательство. Те же рассуждения, что были проведены при доказательстве предложения 1, позволяют убедиться, что для каждого из алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$ арифметическая сложность совпадает по порядку роста с арифметической сложностью (4) алгоритмов EG_σ и $\mathrm{EG}_{\sigma^{-1}}$, описанных в п. 5.3.1.

Прямой проверкой можно убедиться, что сдвиговая сложность алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$ есть $O(n^3d^3)$. Теперь для доказательства (9) достаточно каждой паре n, d сопоставить оператор $L_{(n,d)}$ такой, что при любых n, d для некоторой положительной константы c, общей для всех n и d, число сдвигов, требуемых любым из алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$, больше, чем cn^3d^3. Например, для RR_σ оператор $L_{(n,d)}$ может быть выбран среди таких операторов, у которых верхние порядки первых $[n/2]$ строк равны d, а верхние порядки остальных строк равны $[d/2]$, и при этом верхние порядки всех строк после применения RR равны $[d/2]$. Нужно также, чтобы с каждым шагом применения алгоритма сумма верхних порядков строк уменьшалась ровно на единицу, и чтобы в каждом исключении участвовали все строки, имеющие порядки $[d/2]$. Для $\mathrm{RR}_{\sigma^{-1}}$ верно все то же самое, но вместо верхних рассматриваются нижние порядки.

Таким образом, при $n, d \to \infty$ сдвиговая сложность алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$ имеет более высокий порядок роста, чем сдвиговая сложность алгоритмов EG_σ и $\mathrm{EG}_{\sigma^{-1}}$.

В [1, предложение 6] для дифференциального случая доказано, что если хранить все результаты дифференцирования строк и не повторять уже выполненных дифференцирований, то общее число дифференцирований строк для дифференциальной версии RR будет допускать в худшем

случае оценку $O(nd^2)$. Не утверждается, что эта оценка точна в том или ином смысле. Эта оценка справедлива и для числа сдвигов строк в разностном случае (доказательство, по существу, остается тем же самым). Хранение всех результатов сдвигов строк существенно увеличивает пространственную сложность, но интересующая нас сдвиговая сложность алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$ уменьшается: вместо (9) мы получаем

$$O(n^2d^3). \tag{10}$$

С помощью предложения 2 мы можем оценить сложность алгоритмов ExtRR_σ и $\mathrm{ExtRR}_{\sigma^{-1}}$. Если по-прежнему считать, что $\operatorname{ord} W = d_1$, то мы имеем оценку $\Theta(nd(n^\omega + n^2\cdot(d+d_1))) = \Theta(n^{\omega+1}d + n^3d^2 + n^3dd_1)$ для арифметической сложности и $\Theta(nd\cdot n^2\cdot(d+d_1)^2) = \Theta(n^3d^3 + n^3d^2d_1 + n^3dd_1^2)$ для сдвиговой сложности. Когда все результаты сдвигов сохраняются, для сдвиговой сложности имеем оценку $O(nd\cdot n\cdot(d+d_1)^2) = O(n^2d^3 + n^2d^2d_1 + n^2dd_1^2)$.

Отсюда с учетом замечания 4 получаем следующее

Предложение 4. *При* $\operatorname{ord} W = \Theta(nd)$ *арифметическая и сдвиговая сложности алгоритмов* ExtRR_σ *и* $\mathrm{ExtRR}_{\sigma^{-1}}$ *допускают оценки*

$$\Theta(n^4d^2),\quad \Theta(n^5d^3), \tag{11}$$

если не прибегать к хранению результатов всех сдвигов строк. Когда все результаты сдвигов сохраняются, сложности допускают оценки

$$\Theta(n^4d^2),\quad O(n^4d^3). \tag{12}$$

5.4. Треугольные матрицы

Определение 4. Пусть i-я строка фронтальной матрицы оператора L имеет вид

$$(\underbrace{0, \ldots, 0}_{k-1}, a, \ldots, b),\quad 1 \leqslant k \leqslant n,\quad a \neq 0.$$

Тогда число k будем называть *пин-индексом* i-й строки оператора L. Если i-я строка оператора L нулевая, то ее пин-индекс равен $-\infty$.

Если строки L имеют попарно различные положительные пин-индексы, то фронтальная матрица невырожденна: с точностью до порядка строк она будет треугольной с ненулевыми диагональными элементами. Допустим, что строки $r_1 = L_{i,*}$ и $r_2 = L_{j,*}$ имеют одинаковые положительные пин-индексы, равные k, и при этом $\overline{\operatorname{ord}}\, L_{i,*} = d_1$, $\overline{\operatorname{ord}}\, L_{j,*} = d_2$, $d_1 \leqslant d_2$. Выполнив одну арифметическую операцию в $\mathbb{K}$, можно найти $v \in \mathbb{K}$ такое, что строка

$$r_2 - v\sigma^{d_2-d_1}r_1 \tag{13}$$

имеет или больший, чем k, пин-индекс, или меньший, чем d_2, верхний порядок (возможно и то, и другое вместе). Та из строк r_1, r_2, которая имеет больший порядок, заменяется в L строкой (13); когда порядки этих строк равны, заменяется любая из них. Если L имеет полный ранг, то фронтальная матрица после не более, чем $n\cdot nd$ таких унимодулярных преобразований (каждое из них эквивалентно умножению слева на унимодулярный оператор), станет треугольной. Этот прием может использоваться вместо поиска линейной зависимости строк фронтальной матрицы.

Замечание 5. В предположении, что $\mathbb{K}$ есть поле рациональных функций от x с автоморфизмом $x \to x+1$, этот прием был использован в [12] в первой версии алгоритма EG. В обсуждении дифференциального случая А. Шторйоханн привлек внимание автора к тому, что сложность этого подхода меньше, чем подхода, связанного с решением линейных алгебраических систем (см. также [20]).

Можно также предложить алгоритм $\Delta\mathrm{RR}_{\sigma^{-1}}$ для получения эквивалентного оператора с невырожденной тыльной матрицей.

Для EG_σ ведущая и фронтальные матрицы совпадают в моменты сопоставления пин-индексов, при этом $d_2 - d_1 = 0$ в (13). Строка (13) заменяет ту из строк r_1, r_2, которая имеет меньший

нижний порядок, что эквивалентно выбору для этой замены строки с бóльшим порядком (если порядки строк равны, то берем любую из них).

В алгоритм добавляется предварительный шаг выравнивания верхних порядков всех строк с помощью σ. (Это преобразование унимодулярно, так как σ и σ^{-1} являются обратными по отношению друг к другу.) На этом пути получаем алгоритм $\Delta\mathrm{EG}_\sigma$.

Замечание 6. Если алгоритм $\Delta\mathrm{EG}_\sigma$ применяется к оператору с невырожденной тыльной матрицей, то в результате получим оператор, который, помимо невырожденной ведущей матрицы, будет иметь невырожденную тыльную матрицу (аналогично EG_σ – см. замечание 2). Это является следствием используемого правила замены: "Та из строк r_1, r_2, которая имеет больший порядок, заменяется в L строкой (13); когда порядки этих строк равны, заменяется любая из них". Если же следовать аналогии с дифференциальным случаем [1], то правилом замены должно быть: "Строка r_2 заменяется в L строкой (13)", но тогда сохранение невырожденности для тыльной матрицы могло бы не иметь места (в дифференциальном случае это сохранение не представляет интереса).

Подобным же образом получаем алгоритм $\Delta\mathrm{EG}_{\sigma^{-1}}$. Он находит оператор, эквивалентный исходному, трейлинговая матрица которого не вырождена.

Предложение 5. *Для* ΔEG*-алгоритмов арифметическая сложность допускает оценку*

$$\Theta(n^3d^2). \tag{14}$$

Сдвиговая сложность допускает оценку

$$\Theta(n^2d^2). \tag{15}$$

Доказательство. Очевидно, что для семейства ΔEG арифметическая сложность допускает оценку $\Theta(nd\cdot n\cdot nd)$, т.е. (14). Также легко видеть, что для семейства ΔEG число сдвигов в худшем случае такое же, как для EG, т.е. $\Theta(n^2d^2)$.

6. ПРОВЕРКА УНИМОДУЛЯРНОСТИ, ОБРАЩЕНИЕ ОПЕРАТОРОВ

6.1. Проверка унимодулярности

В конце п. 5.2 упоминался алгоритм проверки унимодулярности, имеющий арифметическую сложность (4) и сдвиговую сложность (5). На основе алгоритмов $\Delta\mathrm{EG}_{\sigma^{-1}}$ и $\Delta\mathrm{EG}_{\sigma-1}$ можно предложить алгоритм с меньшей арифметической сложностью, сохранив прежнюю оценку для сдвиговой сложности. Именно, сначала с помощью $\Delta\mathrm{EG}_{\sigma^{-1}}$ преобразуем исходный оператор L в эквивалентный оператор с невырожденной трейлинговой матрицей. Затем к оператору-результату применим $\Delta\mathrm{EG}_\sigma$. Если попутно не обнаружилось, что исходный оператор не имел полного ранга, то в результате этого применения $\Delta\mathrm{EG}_{\sigma^{-1}}$ и $\Delta\mathrm{EG}_\sigma$ в соответствии с замечанием 6 мы будем знать значение $\dim V_L$. По теореме 1(ii) имеем $L\in\Upsilon_n\Leftrightarrow\dim V_L=0$.

Принимая во внимание оценки (14), (15), мы заключаем, что для описанного алгоритма арифметическая и сдвиговая сложность представляют собой, соответственно,

$$\Theta(n^3d^2),\quad \Theta(n^2d^2). \tag{16}$$

6.2. Построение обратного оператора

Основу для алгоритма обращения дают алгоритмы из п.п. 5.3.2. Сам алгоритм обращения мы представим в виде четырех шагов. Если оператор имеет неполный ранг, то это будет обнаружено на первом шаге и выполнение алгоритма можно будет закончить.

Шаг 1. Применить $\mathrm{ExtRR}_{\sigma^{-1}}$ к L, I_n; пусть результатом будет $L_1\in\mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, $W_1\in\Upsilon_n$.

Шаг 2. Применить ExtRR_σ к L_1, W_1; пусть результатом будет $L_2\in\mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, $W_2\in\Upsilon_n$.

Шаг 3. Если $\mathrm{ord}\,L_2>0$, то $L\notin\Upsilon_n$.

Шаг 4. Пусть $\mathrm{ord}\,L_2=0$ и $\beta_1,\ldots,\beta_n$ – нижние порядки строк оператора L_2. Тогда $L_2=MD$, где $M\in\mathrm{Mat}_n(\mathbb{K})$, $D=\mathrm{diag}(\sigma^{\beta_1},\ldots,\sigma^{\beta_n})$. Отсюда $L^{-1}=\mathrm{diag}(\sigma^{-\beta_1},\ldots,\sigma^{-\beta_n})M^{-1}W_2$.

После выполнения шага 2 в операторе L_2 фронтальная и тыльная матрицы невырожденны. Шаги 1–3 имеют в совокупности арифметическую и сдвиговую сложности (10). Арифметическая сложность вычисления $M^{-1}W_2$ (шаг 4) есть $O(n^2 \cdot nd) = O(n^3 d)$. Эта сложность растет медленнее, чем $n^4 d^2$. Первая из входящих в (10), (12) оценок не меняется. Для получаемых на шаге 4 значений $\beta_1, \ldots, \beta_n$ выполняется $0 \leqslant \beta_i \leqslant d$, $i = 1, \ldots, n$. При этом $\operatorname{ord} M^{-1}W_2 = \operatorname{ord} W_2 \leqslant$ $\leqslant (n-1)\operatorname{ord} L = (n-1)d$. Поэтому сдвиговые затраты в худшем случае для умножения $\operatorname{diag}(\sigma^{-\beta_1}, \ldots, \sigma^{-\beta_n})$ на $M^{-1}W_2$ допускают оценку $O(n^2 d^2)$, а вторые оценки, входящие в (10), (12), не меняются.

В итоге мы установили, что справедлива

Теорема 2. (i) *Для проверки унимодулярности произвольного оператора L существует алгоритм, арифметическая и сдвиговая сложности которого допускают, соответственно, оценки $\Theta(n^3 d^2)$ и $\Theta(n^2 d^3)$. Результатом применения алгоритма является "да"* (*оператор унимодулярен*) *или "нет"* (*оператор неунимодулярен*).

(ii) *Для проверки унимодулярности произвольного оператора L и в случае унимодулярности – построения L^{-1}, существует алгоритм, арифметическая и сдвиговая сложности которого допускают, соответственно, оценки*

$$\Theta(n^4 d^2), \quad O(n^4 d^3). \tag{17}$$

Результатом применения алгоритма служит оператор L^{-1} или сообщение о том, что оператор L не является унимодулярным. (*Если не хранить результаты всех выполненных сдвигов, то арифметическая и сдвиговая сложности будут представлять собой $\Theta(n^4 d^2)$, $\Theta(n^5 d^3)$, но при этом уменьшается пространственная сложность, т.е. расход дополнительной памяти в худшем случае.*)

Пример 2. Оператор из левой части равенства (7) запишем в развернутом виде:

$$L = \begin{pmatrix} 1 & -\dfrac{1}{x}\sigma \\ \dfrac{x^2}{2} & -\dfrac{x}{2}\sigma + 1 \end{pmatrix} = \begin{pmatrix} 0 & -\dfrac{1}{x} \\ 0 & -\dfrac{x}{2} \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ \dfrac{x^2}{2} & 1 \end{pmatrix}.$$

Тыльная матрица совпадает с трейлинговой, и эта последняя невырожденна. Применяем ExtRR_σ. При этом оператор L претерпевает следующие преобразования:

$$\begin{pmatrix} 0 & -\dfrac{1}{x} \\ 0 & -\dfrac{x}{2} \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ \dfrac{x^2}{2} & 1 \end{pmatrix} \to \begin{pmatrix} 0 & -\dfrac{1}{x} \\ 0 & 0 \end{pmatrix}\sigma + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Последняя матрица цепочки является матрицей I_2, она имеет полный ранг и ее порядок равен нулю. Имеем $\beta_1 = \beta_2 = 0$. Отсюда следует, что исходный оператор L унимодулярен. Оператор W, изначально равный I_2, в ходе применения ExtRR_σ преобразуется так:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \to \begin{pmatrix} 1 & 0 \\ -\dfrac{x^2}{2} & 1 \end{pmatrix} \to \begin{pmatrix} 1 - \dfrac{(x+1)^2}{2x}\sigma & \dfrac{1}{x}\sigma \\ -\dfrac{x^2}{2} & 1 \end{pmatrix}. \tag{18}$$

В соответствии с алгоритмом, умножаем последний оператор этой цепочки слева на обратную к I_2 матрицу. Это умножение не изменяет последний оператор цепочки (18), то же самое – умножение на $\operatorname{diag}(\sigma^0, \sigma^0)$. Таким образом, последний оператор цепочки (18) является обратным к L. Он совпадает с оператором в правой части равенства (7).

Что касается 1×1-матриц, то, согласно приведенным выше алгоритмам, каждая такая матрица унимодулярна, если и только если единственный ее элемент a является оператором порядка 0, т.е. $a \in \mathrm{K}\backslash\{0\}$, в этом случае обратная матрица состоит из единственного элемента a^{-1}.

Легко заметить, что в алгоритме обращения оператора мы не использовали подход, основанный на получении треугольных матриц и обсуждавшийся в п. 5.4, хотя этот подход позволил понизить арифметическую сложность алгоритма проверки унимодулярности. По аналогии с ExtRR_σ, $\mathrm{ExtRR}_{\sigma^{-1}}$ мы можем описать алгоритмы $\mathrm{Ext}\Delta\mathrm{RR}_\sigma$, $\mathrm{Ext}\Delta\mathrm{RR}_{\sigma^{-1}}$, но остается неясным, например, будут ли на всех шагах этих новых алгоритмов выполняться утверждения предложения 2.

7. ПОЛЕ РАЦИОНАЛЬНЫХ ФУНКЦИЙ В РОЛИ ИСХОДНОГО РАЗНОСТНОГО ПОЛЯ

Если элементы поля $\mathbb{K}$ представлены выражениями, в которые входит переменная x, а сдвиг σ есть замена x на $x+1$ с возможным последующим упрощением выражения, то при любом $k \in \mathbb{Z}$ затраты на применение σ^k и σ можно считать одинаковыми. Это изменяет в меньшую сторону сдвиговую сложность некоторых из рассмотренных выше алгоритмов. Мы займемся исследованием этих изменений, считая, что поле $\mathbb{K}$ – это поле $K(x)$ рациональных функций над некоторым полем K характеристики 0 и $\sigma x = x+1$.

Здесь нелишне отметить, что если применение σ^k к какому-то $a \in \mathbb{K}$ потребовало нахождения $\sigma a, \ldots, \sigma^k a$, то эти значения можно запомнить для дальнейшего использования. В то же время такое вычисление $\sigma^k a$, как мы обсуждаем здесь, не даст нам $\sigma a, \ldots, \sigma^{k-1} a$, и если нам в дальнейшем понадобится, например, элемент $\sigma^{k-1} a$, то потребуется его вычисление. Поэтому сложность какого-либо алгоритма при обсуждаемом здесь взгляде на связанные с применением σ^k затраты не обязательно должна быть меньше, чем при той трактовке сдвиговой сложности, которой мы придерживались в предыдущих разделах. Но в нашей ситуации некоторое уменьшение сдвиговой сложности все-таки имеет место.

Легко устанавливается, что для алгоритмов RR_σ и $\mathrm{RR}_{\sigma^{-1}}$ оценка (9) преобразуется в $\Theta(n^3 d^2)$: каждый из nd шагов в худшем случае требует $n-1$ сдвигов строк. Для алгоритмов ExtRR_σ и $\mathrm{ExtRR}_{\sigma^{-1}}$ вторая из оценок (10), касающаяся сдвиговой сложности, преобразуется в $\Theta(n^4 d^2)$.

Для семейств ΔEG и ΔRR оценка (15) остается неизменной, в то же время оценка для сдвиговой сложности алгоритмов семейства ΔRR становится такой же, как для ΔEG: при новом взгляде на сдвиги вида σ^k каждый шаг требует в худшем случае n сдвигов строк по nd элементов в каждой.

Оценка сдвиговой сложности для алгоритма проверки унимодулярности (вторая оценка, входящая в (16)) остается неизменной, но в соответствующей оценке сдвиговой сложности для построения обратного оператора происходит замена d^3 на d^2 по причине аналогичных изменений в сдвиговой сложности алгоритмов ExtRR_σ и $\mathrm{ExtRR}_{\sigma^{-1}}$. Таким образом, (17) преобразуется в $\Theta(n^4 d^2)$, $O(n^4 d^2)$.

Нами доказана

Теорема 3. *Пусть $\mathbb{K} = K(x)$, где K – поле характеристики 0 и x, переменная. Пусть $\sigma R(x) = R(x+1)$ для любой рациональной функции $R(x) \in K(x)$. Тогда существует такой алгоритм проверки унимодулярности произвольного оператора и построения обратного оператора в случае его существования, что для как арифметической, так и сдвиговой сложности этого алгоритма справедлива оценка $O(n^4 d^2)$.*

Аналогичный результат можно получить для q-разностного случая. В наиболее простом его варианте $\mathbb{K} = K(q, x)$, где q – еще одна переменная и автоморфизм σ определяется посредством $x \to qx$.

8. ДРУГИЕ ПОДХОДЫ

Связанные с проверкой унимодулярности и построением обратной матрицы задачи могут решаться разными алгоритмами. Например, может использоваться построение форм Джекобсона и Эрмита заданной операторной матрицы; определения этих форм можно найти в [21], [22]. В [21] сложность предложенного там алгоритма построения формы Джекобсона рассмотрена

как функция трех переменных, и две из них – это наши n, d (в [21] привлекались другие обозначения). Значение третьей переменной в худшем случае равно nd, и для сложности как функции переменных n, d можно получить оценку $\Theta(n^9 d^9)$. Формой Эрмита унимодулярной матрицы служит единичная матрица, и в этом случае матрица U преобразования является обратной для исходной матрицы. Приведенная в [22] оценка сложности имеет в наших обозначениях вид $O(n^7 d^3 \log(nd))$. Похоже, что последняя оценка является точной. (Конечно, алгоритмы из [21], [22] решают более общие задачи, и обсуждавшиеся выше алгоритмы имеют некоторые преимущества только для проверки унимодулярности и построения обратной операторной матрицы.) В [21], [22] алгоритмы описаны на языке колец некоммутативных полиномов Оре (см. [23], [24]), что делает эти алгоритмы применимыми в дифференциальном и разностном случаях.

Заметим, что мы обсуждаем алгоритмы со сложностной точки зрения. Алгоритм, который выглядит лучшим в этом смысле, не обязательно должен оказаться лучшим в вычислительной практике.

9. ОТКРЫТЫЕ ВОПРОСЫ, ГИПОТЕЗЫ

Неясно, существует ли алгоритм проверки унимодулярности со сложностью, допускающей оценку $O(n^\alpha d^\beta)$, где α, β суть вещественные числа и $\alpha < 3$. Для матриц, элементами которых являются обычные коммутативные полиномы из $K[x]$, известен опубликованный в [25] алгоритм построения обратной матрицы со сложностью $O(n^3\rho)$, где ρ является максимальной степенью элементов данных матриц (строго говоря, алгоритм из [25] предназначен для обращения только матриц “общего положения”). Неясно также, сводится ли задача построения обратной операторной матрицы к задаче умножения матриц, аналогично сводимости в случае, когда элементы матриц принадлежат полю (см. [26, разд. 16.4], [27, гл. 6]). Сводимость здесь понимается в том смысле, что если существует алгоритм умножения операторных матриц со сложностью (арифметической или сдвиговой) $T(n, d)$, то существует алгоритм обращения со сложностью $O(T(n, d))$. Предположение об этой сводимости вызывает сомнения, но при этом сводимость в другую сторону доказывается так же, как для матриц над полем.

Возвращаясь к матрицам с полиномиальными элементами, заметим, что существует алгоритм умножения матриц со сложностью $O(n^\omega \rho f(\log n \log \rho))$, где ω, как и прежде – показатель матричного умножения, $2 < \omega \leqslant 3$, ρ вновь обозначает максимальную степень элементов данных матриц, f – некоторый полином (см. [28]). Однако алгоритм с такой сложностью для обращения матриц, скорее всего, не существует.

Но сказанное – это не более, чем предположение. Автор не располагает сведениями об алгоритмах, которые бы решали, например, задачу проверки унимодулярности с меньшей, чем указана в теореме 2(i), сложностью. Поиск в литературе не дал положительного результата, но, конечно, это не исключает существования такого алгоритма. Возможно, например, что с использованием идей, на которых построены алгоритмы быстрого матричного умножения над полем (см. [29]–[31]), и идей алгоритмов быстрого умножения скалярных линейных операторов (см. [32], [19], [33]), можно предложить алгоритм для быстрого умножения операторных матриц, а затем получить соответствующий алгоритм для проверки унимодулярности.

Многие недавние работы (см., например, [18], [22]) направлены на выяснение роста размера принадлежащих $\mathbb{K}$ коэффициентов, когда, например, $\mathbb{K} = \mathbb{Q}(x)$. Было бы интересно исследовать битовую сложность алгоритмов проверки унимодулярности. Другой путь – рассматривать сложность как функцию трех переменных: n, d и ρ, где ρ таково, что все полиномы, участвующие в L как числители и знаменатели коэффициентов элементов, имеют степени, не превосходящие ρ, при этом сложность является для фиксированных n, d и ρ числом операций в поле $\mathbb{Q}$ в худшем случае.

Автор благодарен рецензенту за советы и замечания, касающиеся первого варианта статьи.

СПИСОК ЛИТЕРАТУРЫ

1. *Abramov S.A.* On the differential and full algebraic complexities of operator matrices transformations // LNCS. Heidelberg: Springer, 2016. V. 9890. P. 1–14.

2. *van der Put M., Singer M.F.* Galois theory of differential equations. Grundlehren der mathematischen Wissenschaften. Heidelberg: Springer, 2003. V. 328.
3. *Хованский А.Г.* Топологическая теория Галуа. Разрешимость и неразрешимость уравнений в конечном виде. М.: МЦНМО, 2008.
4. *van der Put M., Singer M.F.* Galois theory of difference equations. LNM, Heidelberg: Springer, 1997. V. 1666.
5. *Abramov S., Barkatou M.* On the dimension of solution spaces of full rank linear differential systems. LNCS. Heidelberg: Springer, 2016. V. 8136. P. 1–9.
6. *Абрамов С.А.* Об обращении разностных операторных матриц // В сб. Дифференциальные уравнения и смежные вопросы математики. Тр. VIII Приокской научн. конференции, 10–11 июня 2016 года. Коломна-Константиново. 2016. С. 4–12.
7. *Кац В.Г., Чен П.* Квантовый анализ. М: МЦНМО, 2005.
8. *Andrews G.E.* q-Series: their development and application in analysis, number theory, combinatorics, physics, and computer algebra. Pennsylvania: CBMS Regional Conference Series, AMS, R.I., 1986. V. 66.
9. *Abramov S., Barkatou M.* On solution spaces of products of linear differential or difference operators // ACM Comm. in Computer Algebra, 2014. V. 4. P. 155–165.
10. *Franke C.H.* Picard-Vessiot theory of linear homogeneous difference equations // Trans. Amer. Math. Soc. 1963. V. 108. P. 491–515.
11. *Knuth D.E.* Big omicron and big omega and big theta // ACM SIGACT News. 1976. V. 8. № 2. P. 18–23.
12. *Abramov S.* EG-eliminations // J. of Difference Equat. Applicat. 1999. V. 5. P. 393–433.
13. *Abramov S., Bronstein M.* Linear algebra for skew-polynomial matrices // Rapport de Recherche INRIA, March 2002, RR-4420. http://www.inria.fr/RRRT/RR-4420.html.
14. *Abramov S., Bronstein M.* On solutions of linear functional systems // ISSAC'2001 Proc. 2001. P. 1–6.
15. *Абрамов С.А., Хмельнов Д.Е.* Линейные дифференциальные и разностные системы: EG_δ- и EG_σ-исключения // Программирование. 2013. № 2. С. 51–74.
16. *Barkatou M.A., El Bacha C., Labahn G., Pflügel E.* On simultaneous row and column reduction of higher-order linear differential systems // J. Symbolic Comput. 2013. V. 49. № 1. P. 45–64.
17. *Beckermann B., Labahn G.* Fraction-free computation of matrix rational interpolants and matrix GCDs // SIAM J. Matrix Anal. Appl. 2000. V. 77. № 1. P. 114–144.
18. *Beckermann B., Cheng H., Labahn G.* Fraction-free row reduction of matrices of Ore polynomials // J. Symbolic Comput. 2006. V. 41. P. 513–543.
19. *Benoit A., Bostan A., van der Hoeven J.* Quasi-optimal multiplication of linear differential operators // Proc. FOCS'12. New Brunswick: IEEE Comp. Soc. Press. 2012. P. 524–530.
20. *Mulders T., Storjohann A.* On lattice reduction for polynomial matrices // J. Symb. Comput. 2004. V. 37. № 4. P. 485–510.
21. *Middeke J.* A polynomial-time algorithm for the Jacobson form for matrices of differential operators. Tech. Report in RISC Report Series. 2008. № 08-13.
22. *Giesbrecht M., Sub Kim M.* Computation of the Hermite form of a Matrix of Ore Polynomials // J. Algebra. 2013. V. 376. P. 341–362.
23. *Ore O.* Theory of non-commutative polynomials // Annals of Mathematics. 1933. V. 34. P. 480–508.
24. *Bronstein M., Petkovšek M.* An introduction to pseudo-linear algebra // Theor. Comput. Sci. 1996. V. 157. № 1. P. 3–33.
25. *Jeannerod C.-P., Villard G.* Essentially optimal computation of the inverse of generic polynomial matrices // J. Complexity. 2005. V. 21. № 1. P. 72–86.
26. *Bürgisser P., Clausen M., Shokrollahi M.A.* Algebraic complexity theory. Grundlehren der mathematischen Wissenschaften, Heidelberg: Springer, 1997. V. 315.
27. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.
28. *Bostan A., Schost E.* Polynomial evaluation and interpolation on special sets of points // J. Complexity. 2005. V. 21. № 4. P. 420–446.
29. *Strassen V.* Gaussian elimination is not optimal // Numer. Math. 1969. V. 13. P. 354–356.
30. *Coppersmith D., Winograd S.* Matrix multiplication via arithmetic progressions // J. Symb. Comput. 1990. V. 9. № 3. P. 251–280.
31. *Vassilevska Williams, V.* Multiplying matrices faster than Coppersmith–Winograd // STOC'2012 Proc. P. 887–898.
32. *van der Hoeven J.* FFT-like multiplication of linear differential operators // J. Symb. Comput. 2002. V. 33. P. 123–127.
33. *Bostan A., Chyzak F., Le Roix N.* Products of ordinary differential operators by evaluating and interpolation // Proc. ISSAC'2008. 2008. P. 23–30.