УДК 004.421.6

ПАКЕТ ПРОЦЕДУР ОБРАЩЕНИЯ МАТРИЦ С ЛИНЕЙНЫМИ РАЗНОСТНЫМИ ОПЕРАТОРАМИ В РОЛИ ЭЛЕМЕНТОВ*

© 2018 г. С. А. Абрамов, Д. Е. Хмельнов

Федеральный исследовательский центр "Информатика и управление" РАН

119333 Москва, ул. Вавилова, 40

E-mail: sergeyabramov@mail.ru, dennis_khmelnov@mail.ru

Поступила в редакцию 31.08.2018

Рассматриваются матрицы, принадлежащие $\mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$, т.е. кольцу $n\times n$-матриц, элементами которых являются скалярные разностные операторы с коэффициентами в разностном поле $\mathbb{K}$ характеристики 0 с автоморфизмом ("сдвигом") σ. Обсуждается некоторое семейство алгоритмов, позволяющих проверить, существует ли для данной матрицы из $\mathrm{Mat}_n(\mathbb{K}[\sigma,\sigma^{-1}])$ обратная в этом кольце, и если существует, то позволяющих также и построить обратную матрицу. Этим алгоритмам сопоставляются сложности по числу арифметических операций и по числу сдвигов (т.е. применений σ и σ^{-1}) в поле $\mathbb{K}$. Алгоритмы реализованы в виде `Maple`-процедур. Это дает возможность экспериментального сравнения затрачиваемого времени. Выбор лучшего алгоритма на основе этих экспериментов не всегда совпадает с выбором на сложностной основе. Делается попытка выяснить причины этого несовпадения. Предлагается пакет процедур решения рассматриваемых задач; в главной процедуре можно использовать параметр, указывающий на один из реализованных алгоритмов. В отсутствие этого параметра выбирается некоторый фиксированный алгоритм, достаточно удачно выглядящий как в сложностном, так и в экспериментальном сравнении с другими.

1. ВВЕДЕНИЕ

Матрицы с элементами, принадлежащими самым разным кольцам и полям, используются во всех областях математики и в приложениях. В этой статье речь идет о матрицах, элементы которых принадлежат кольцу линейных разностных скалярных операторов над некоторым разностным полем $\mathbb{K}$ с автоморфизмом (сдвигом) σ. Предполагается, что поле $\mathbb{K}$ имеет характеристику 0. Изложение разворачивается вокруг задач проверки обратимости данной матрицы обсуждаемого типа и нахождения обратной матрицы, сколь скоро она существует. Мы даем беглый обзор известных алгоритмов, основанных на регуляризации, т.е. получении в невырожденном виде сопоставляемых исходной операторной матрице ведущих и трейлинговых матриц, элементы которых принадлежат $\mathbb{K}$. В некоторых алгоритмах регуляризируются не ведущая и трейлинговая, а фронтальная и тыльная матрицы (их элементы также принадлежат $\mathbb{K}$; в [1] использован общий термин "выявляющие матрицы").

Таким образом, для матриц обсуждаемого вида уже предложено некоторое число алгоритмов обращения, и мы приводим перечень этих алгоритмов в пп. 3, 4. Для них проведено исследование сложности, где сложность рассматривается в двух вариантах:

1) арифметическая сложность, т.е. сложность в худшем случае по числу арифметических операций в поле $\mathbb{K}$,

2) сдвиговая сложность, — тоже в худшем случае, но по числу применений σ и σ^{-1}.

Некоторые алгоритмы этого множества уже были реализованы в виде `Maple`-процедур. Мы предоставляем недостающие реализации тех алгоритмов, которые имеют сравнительно небольшие сложности и проводим экспериментальное сравнение этих алгоритмов по фактическим затратам времени. Интересно, что алгоритмы, имеющие меньшую сложность, не всегда оказываются самыми быстрыми, и это, разумеется, не является ни в какой мере парадоксом, так как сложность, будь то арифметическая или сдвиговая, или даже сложность по общему числу тех и других операций в худшем случае, во-первых, не характеризует всех затрат на выполнение алгоритма, и, во-вторых, сравнение затрат в худшем случае не охватывает всех мыслимых случаев, а с "худшим случаем" мы можем сталкиваться весьма редко, и охватываемые набором тестов случаи могут не попадать в разряд худших. Нелишне еще заметить, что когда мы сравниваем сложности, исходя из их асимптотических оценок, то не всегда получаем адекватное представление о значениях той или иной сложности для данных не слишком большого (не "астрономического") размера.

*Частичная поддержка РФФИ, грант 19-01-00032-а.

Для выбора наиболее удобного, например, — наиболее быстрого, алгоритма из некоторого множества, сложностные характеристики алгоритмов этого множества играют, естественно, немаловажную роль. Но если сложности разных алгоритмов являются близкими величинами, для которых мы располагаем лишь асимптотическими оценками, то эти оценки дадут нам возможность лишь предварительного отбора. Последующее экспериментальное сравнение (тестирование) оказывается очень желательным, особенно когда тестовые задания имеют размеры "типичные" для тех ситуаций, на которые ориентирована реализация алгоритмов.

Итак, в статье мы даем как некоторые сложностные оценки для алгоритмов рассматриваемого множества, так и результаты экспериментального сравнения этих алгоритмов.

Здесь надо сказать, что в [2] мы указывали ссылку на страницу в интернете, содержащую реализацию алгоритма с наименьшими в асимптотическом смысле арифметической и сдвиговой сложностью. Отмечалось, что в реализации было сделано некоторое отступление от самого алгоритма в части регуляризации ведущей и трейлинговой матриц с помощью некоторого специального варианта алгоритма EG ([3, 4]). Однако выбор подходящего варианта этой регуляризации оставлен в [2], фактически, "за кадром". В настоящей статье сравнению алгоритмов на уровне реализации уделено особое внимание.

Предлагается пакет процедур решения рассматриваемых задач; в главной процедуре можно использовать параметр, указывающий на один из реализованных алгоритмов. В отсутствие этого параметра выбирается некоторый фиксированный алгоритм, достаточно удачно выглядящий как в сложностном, так и в экспериментальном сравнении с другими.

Завершим это введение в статью несколькими примечаниями и соглашениями.

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 $\delta = '$ и когда элементы матриц суть скалярные линейные дифференциальные операторы над $\mathbb{K}$, рассматривались в [5]. Для данной операторной матрицы L как дифференциальные, так и обсуждаемые ниже разностные алгоритмы вычисляют размерность пространства решений V_L соответствующей системы уравнений $L(y) = 0$. Делается это в предположении, что компоненты решений принадлежат связанному с L адекватному (см. п.2) расширению поля $\mathbb{K}$. Как было установлено ранее (см. [6]), 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.

Будем в дальнейшем придерживаться следующих стандартных обозначений. Кольцо $n \times n$-матриц (n — положительное целое) с элементами в некотором кольце или поле R обозначается через $\mathrm{Mat}_n(R)$. Если M — некоторая $n \times n$-матрица, то обозначение $M_{i,*}$, $1 \leqslant i \leqslant n$, 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.

2. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

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*. 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 $\mathbb{K}$ (соответствующий автоморфизм кольца Λ совпадает на $\mathbb{K}$ с σ, для этого автоморфизма кольца Λ используется то же самое обозначение σ).

Определение 1. 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

$$\sigma y = Ay, \;\; y = (y_1, \ldots, y_n)^T,$$

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

Существование некоторого адекватного разностного расширения Λ для произвольного разностного поля доказывается достаточно элементарно — см. [7, п.5.1]. Для произвольного 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.

Примечание 1. Так называемый q-разностный случай ([8, 9]) охватывается общим разностным случаем.

Скалярный разностный оператор является элементом кольца $\mathbb{K}[\sigma, \sigma^{-1}]$.

Определение 2. Для ненулевого скалярного оператора $f = \sum a_i \sigma^i$ мы определяем его *ведущий и трейлинговый порядки*:

$$\overline{\operatorname{ord}}\, f = \max\{i \mid a_i \neq 0\}, \;\; \underline{\operatorname{ord}}\, f = \min\{i \mid a_i \neq 0\},$$

а также *порядок* $\operatorname{ord} f = \overline{\operatorname{ord}}\, f - \underline{\operatorname{ord}}\, f$. Полагаем $\overline{\operatorname{ord}}\, 0 = -\infty$, $\underline{\operatorname{ord}}\, 0 = \infty$, $\operatorname{ord} 0 = -\infty$.

Для конечного набора F скалярных операторов (вектора, матрицы, строки матрицы и т.д.) определим ведущий порядок $\overline{\operatorname{ord}}\, F$ как максимум ведущих порядков его элементов, трейлинговый порядок $\underline{\operatorname{ord}}\, F$ — как минимум трейлинговых порядков его элементов и полагаем $\operatorname{ord} F = \overline{\operatorname{ord}}\, F - \underline{\operatorname{ord}}\, F$.

Разностная операторная матрица — это матрица, принадлежащая $\mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$. Далее мы сопоставляем такой операторной матрице некоторые матрицы, принадлежащие $\mathrm{Mat}_n(\mathbb{K})$. Мы будем коротко называть операторную матрицу просто *оператором*.

Оператор имеет *полный ранг* (или: является оператором полного ранга), если его строки линейно независимы над $\mathbb{K}[\sigma, \sigma^{-1}]$.

Если

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

и $L \neq 0$, то L можно записать в *развернутом* виде

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

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 3. Пусть ведущие порядки строк оператора L равны $\alpha_1, \ldots, \alpha_n$, а трейлинговые — $\beta_1, \ldots, \beta_n$. *Фронтальной* матрицей оператора L назовем ведущую матрицу оператора PL, где

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

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

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

Мы говорим, что оператор L *строго редуцирован*, если невырождены как фронтальная, так и тыльная матрицы этого оператора.

Определение 4. Оператор $L \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ называется *унимодулярным* или *обратимым*, если для него существует обратный $L^{-1} \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$: $LL^{-1} = L^{-1}L = I_n$. Множество унимодулярных $n \times n$-операторов будет обозначаться через Υ_n. Два оператора $L_1, L_2 \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ назовем *эквивалентными*, если $L_1 = UL_2$ для некоторого $U \in \Upsilon_n$.

Далее мы обозначаем через Λ некоторое фиксированное адекватное разностное расширение исходного разностного поля $\mathbb{K}$ с автоморфизмом σ. Обозначение V_L привлекается для пространства принадлежащих Λ^n решений системы $L(y) = 0$. Для краткости будем иногда говорить о V_L как о пространстве *решений оператора* L.

Теорема 1. ([10]) *Пусть* $L \in \mathrm{Mat}_n(K[\sigma, \sigma^{-1}])$ *имеет полный ранг. Тогда*

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

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

3. РЕГУЛЯРИЗАЦИЯ: СЕМЕЙСТВА АЛГОРИТМОВ EG И RR

3.1. *Алгоритмы* EG_σ *и* $\mathrm{EG}_{\sigma^{-1}}$

Для L полного ранга алгоритм EG_σ (см. [3, 4, 11]) строит такой эквивалентный разностный оператор $L_+ \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$, что $\overline{\operatorname{ord}}\, L_+ = \overline{\operatorname{ord}}\, L$, $\underline{\operatorname{ord}}\, L_+ \geqslant \underline{\operatorname{ord}}\, L$ и L_+ имеет невырожденную ведущую матрицу. Если ранг L не полон, то алгоритм сообщает об этом.

Не входя в рассмотрение мелких деталей, представим основную идею этого алгоритма. Алгоритм EG_σ строит L_+ на месте L, т.е. L изменяется шаг за шагом, постепенно превращаясь в L_+. На каждом шаге алгоритма проверяется, являются ли строки ведущей матрицы линейно зависимыми над $\mathbb{K}$, и, если являются, то из строк оператора, которым соответствуют ненулевые коэффициенты зависимости, выбирается имеющая наибольший порядок (если таких строк больше одной, то берется любая из них). Затем выбранная строка редуцируется с помощью остальных строк оператора. Если получается нулевая строка, то исходный оператор не имел полного ранга. Иначе с помощью σ редуцированная строка сдвигается так, чтобы ее ведущий порядок сравнялся с ведущими порядками других строк оператора.

Показано, что после нескольких шагов алгоритм завершается: либо получаем нулевую строку в операторе либо приходим к оператору с невырожденной ведущей матрицей.

Аналогично, алгоритм $\mathrm{EG}_{\sigma^{-1}}$ строит эквивалентный разностный оператор L_- с невырожденной трейлинговой матрицей.

Расширенный алгоритм EG_σ (мы назовем его ExtEG_σ) позволяет наряду с L_+ найти такой унимодулярный оператор $U \in \Upsilon_n$, что $L_+ = UL$. Можно соответствующим образом определить и алгоритм $\mathrm{ExtEG}_{\sigma^{-1}}$.

Предложение 1. ([2, 10]) *Арифметическая сложность каждого из алгоритмов* EG_σ *и* $\mathrm{EG}_{\sigma^{-1}}$ *допускает оценку*[1]

[1] В тех или иных асимптотических оценках сложности как

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

где ω — показатель матричного умножения, $2 < \omega \leqslant 3$. Для сдвиговой сложности имеет место оценка

$$\Theta(n^2d^2).$$

Арифметическая и сдвиговая сложности алгоритмов ExtEG_σ *и* $\mathrm{ExtEG}_{\sigma^{-1}}$ *допускают оценки*

$$\Theta(n^4d^2),\ \ \Theta(n^4d^2).$$

3.2. *Алгоритмы* RR_σ *и* $\mathrm{RR}_{\sigma^{-1}}$

Для L полного ранга алгоритм RR_σ (см. [14, 15]) строит такой эквивалентный оператор $\breve{L}_+ \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$, что $\overline{\mathrm{ord}}\,\breve{L}_+ \leqslant \overline{\mathrm{ord}}\,L$, $\underline{\mathrm{ord}}\,\breve{L}_+ \geqslant \underline{\mathrm{ord}}\,L$, $\breve{L}_+$ имеет невырожденную фронтальную матрицу. Если ранг L не полон, то алгоритм сообщает об этом.

В основе алгоритма RR_σ также лежит идея поиска линейной зависимости строк матрицы (но не ведущей, а фронтальной) и выполнения сдвигов.

Расширенный алгоритм ExtRR_σ позволяет наряду с $\breve{L}_+$ найти такой унимодулярный оператор $U \in \Upsilon_n$, что $\breve{L}_+ = UL$.

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

Расширенный вариант алгоритма $\mathrm{RR}_{\sigma^{-1}}$ мы назовем $\mathrm{ExtRR}_{\sigma^{-1}}$.

Предложение 2. ([10]) *Арифметическая сложность каждого из алгоритмов* RR_σ *и* $\mathrm{RR}_{\sigma^{-1}}$ *допускает оценку*

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

Для сдвиговой сложности имеет место оценка

$$\Theta(n^3d^3), \tag{1}$$

если не прибегать к хранению результатов всех сдвигов строк. Если же все результаты сдвигов сохраняются, то (1) заменяется на $\Theta(n^2d^3)$.

Арифметическая и сдвиговая сложности алгоритмов ExtRR_σ *и* $\mathrm{ExtRR}_{\sigma^{-1}}$ *допускают оценки*

$$\Theta(n^4d^2),\ \ \Theta(n^5d^3),$$

если не хранить результаты всех сдвигов строк. Когда эти результаты сохраняются, сложности допускают оценки

$$\Theta(n^4d^2),\ \ O(n^4d^3).$$

3.3. *Алгоритмы* $\triangle\mathrm{EG}_\sigma$, $\triangle\mathrm{EG}_{\sigma^{-1}}$

Определение 5. Пусть i-я строка ведущей матрицы оператора L имеет вид

$$(\underbrace{0, \ldots, 0}_{k-1}, a, \ldots, b),$$

$1 \leqslant k \leqslant n$, $a \neq 0$. Тогда число k будем называть *отступом* i-й строки оператора L. Если i-я строка оператора L нулевая, то ее отступ равен $-\infty$.

Если строки L имеют попарно различные положительные отступы, то ведущая матрица невырождена: 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 отступы, равные k, и при этом $\overline{\mathrm{ord}}\,L_{i,*} = \overline{\mathrm{ord}}\,L_{j,*} = d$. Выполнив одну арифметическую операцию в $\mathbb{K}$, можно найти $v \in \mathbb{K}$ такое, что строка

$$r_2 - vr_1 \tag{2}$$

имеет или больший, чем k, отступ, или меньший, чем d, ведущий порядок (возможно и то, и другое вместе). Та из строк r_1, r_2, которая имеет меньший трейлинговый порядок, заменяется в L строкой (2); когда порядки этих строк равны, заменяется любая из них. Если L имеет полный ранг, то ведущая 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 ведущей матрицы.

Примечание 2. В предположении, что $\mathbb{K}$ есть поле рациональных функций от x с автоморфизмом $x \to x+1$, этот прием был использован в [3] в первой версии алгоритма EG (см. также [16]).

Использование этого приема приводит к алгоритмам $\triangle\mathrm{EG}_\sigma$ $\triangle\mathrm{EG}_{\sigma^{-1}}$ для получения эквивалентного оператора с невырожденной ведущей или соответственно трейлинговой матрицей.

Примечание 3. Если один из алгоритмов EG_σ, $\triangle\mathrm{EG}_\sigma$ применяется к оператору с невырожденной тыльной матрицей, то в результате получим оператор, который помимо невырожденной ведущей матрицы будет иметь невырожденную тыльную матрицу. Это является следствием используемого правила замены "Та из строк r_1, r_2, которая имеет меньший нижний порядок, заменяется в L строкой (2); когда нижние порядки этих строк равны, заменяется любая из них".

Предложение 3. ([2]) *Для $\triangle$EG-алгоритмов арифметическая сложность допускает оценку*

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

функции переменных n и d (предполагаем, что $n, d \to \infty$) мы прибегаем помимо O-нотации также и к Θ-нотации (см. [12], [13, §2]); соотношение $f(n,d) = \Theta(g(n,d))$ равносильно конъюнкции

$$f(n,d) = O(g(n,d))\ \&\ g(n,d) = O(f(n,d)),$$

проще говоря, $f(n,d)$ и $g(n,d)$ суть величины одного порядка. Соотношение $f(n,d) = \Theta(g(n,d))$ сильнее соотношения $f(n,d) = O(g(n,d))$.

Сдвиговая сложность допускает оценку

$$\Theta(n^2d^2). \tag{4}$$

Для $\mathrm{Ext}\triangle\mathrm{EG}_\sigma$, $\mathrm{Ext}\triangle\mathrm{EG}_{\sigma^{-1}}$ *как арифметическая, так и сдвиговая сложность допускает оценку*

$$O(n^4d^2).$$

Примечание 4. В дифференциальном случае EG-исключения, вообще говоря, приводят к оператору, неэквивалентному исходному. Получаемый оператор имеет все решения, которые имел исходный оператор, но, возможно, имеет и некоторые дополнительные решения. Это послужило причиной того, что в дифференциальном случае как алгоритм проверки унимодулярности, так и алгоритм построения обратного оператора были основаны в [5] на RR_δ. Первый разностный алгоритм ([10]) был аналогичен по своей конструкции дифференциальному алгоритму. Именно последующий переход к алгоритмам EG_σ, $\mathrm{EG}_{\sigma^{-1}}$ как основе алгоритмов проверки унимодулярности и построения обратного оператора позволил в разностном случае понизить сдвиговую сложность.

4. ПРОВЕРКА УНИМОДУЛЯРНОСТИ, ОБРАЩЕНИЕ ОПЕРАТОРОВ

В основе имеющихся алгоритмов проверки унимодулярности оператора L и построения обратного оператора L^{-1} в случае его существования, лежит вычисление $\dim V_L$. Такое вычисление, в свою очередь, основывается на алгоритмах регуляризации (см. п. 3 и примечание 3).

Эквивалентные оператору L операторы

$$\mathrm{EG}_\sigma(\mathrm{EG}_{\sigma^{-1}}(L)),\ \ \triangle\mathrm{EG}_\sigma(\triangle\mathrm{EG}_{\sigma^{-1}}(L)) \tag{5}$$

в силу примечания 3 являются строго редуцированным. По теореме 1 мы можем вычислить $\dim V_L$ и установить, является ли L унимодулярным. Если $\dim V_L = 0$, то в силу теоремы 1(i) соответствующий оператор из (5) принадлежит $\mathrm{Mat}_n(\mathbb{K})$, и L^{-1} вычисляется легко.

Алгоритмы из [10] разработаны по образцу алгоритмов для дифференциального случая [5] с использованием RR, в то время как алгоритм из [2] построен на $\triangle$EG, это позволило снизить сдвиговую сложность— см. примечание 4. В [2] предложены алгоритмы Unimodularity Testing и Inverse Operator, сложности которых (арифметическая и сдвиговая) суть (3), (4) и соответственно

$$O(n^4d^2),\ \ O(n^3d^2).$$

Построение обратного оператора выполняется с привлечением расширенных версий основных алгоритмов, в то время как проверка унимодулярности обходится основными версиями этих же алгоритмов.

Как показано в [10], основанные на RR алгоритмы имеют сложности $\Theta(n^4d^2)$ и $\Theta(n^4d^3)$ (если сохранять результаты всех сдвигов, иначе сдвиговая сложность будет $\Theta(n^4d^5)$. Для алгоритмов, основанных на EG, сложности равны $\Theta(n^4d^2)$, $\Theta(n^4d^2)$).

5. РЕАЛИЗАЦИЯ, ЭКСПЕРИМЕНТАЛЬНОЕ СРАВНЕНИЕ

Реализация рассматриваемых алгоритмов[2] выполнена в среде `Maple` [17] (предварительная версия реализации представлена в [2]). За основу взята описанная в [1] реализация алгоритмов EG, $\triangle$EG, RR для дифференциального случая. Процедуры переработаны для использования в разностном случае, и дополнены реализацией расширенных версий алгоритмов. Также реализован алгоритм проверки унимодулярности оператора и построения обратного оператора, использующий в качестве базового алгоритма один из алгоритмов EG, $\triangle$EG, RR по выбору пользователя. В этой реализации сдвиг определен как $\sigma y(x) = y(x-1)$ и требуется специальный выбор заменяемого уравнения при выполнении шага исключения в алгоритмах EG, $\triangle$EG для гарантии завершения работы. Существует вариант этих алгоритмов, который не требует такого специального выбора заменяемого уравнения. В этом варианте для шага сдвига используется оператор $\sigma - 1$ (использование аналогичного варианта алгоритма EG для q-разностного случая упомянуто в [18, примеч.3]). Но при использовании алгоритмов EG, $\triangle$EG для реализации алгоритма проверки унимодулярности оператора и поиска обратного оператора существенно использование именно версии с контролем порядка исключений (см. примечание 3).

Алгоритмы реализованы в виде процедур нового пакета `EGRRext`. Пакет предоставляет процедуры:

- `EG` – реализует алгоритм EG и его расширенную версию ExtEG;
- `RR` – реализует алгоритм RR и его расширенную версию ExtRR;
- `TriangleEG` – реализует алгоритм $\triangle$EG и его расширенную версию Ext$\triangle$EG;
- `IsUnimodular` – реализует алгоритмы Unimodularity Testing и Inverse Operator.

Оператор $L = A_l\sigma^l + A_{l-1}\sigma^{l-1} + \cdots + A_t\sigma^t$ во входных параметрах процедур представляется в виде списка

$$[A, l, t],$$

где A – *явная матрица*

$$A = (A_l|A_{l-1}|\ldots|A_t)$$

[2]Доступна по адресу http://www.ccas.ru/ca/egrrext

размера $n \times n(l-t+1)$. Явная матрица A представляется в виде стандартного Maple-объекта Matrix. Элементами явной матрицы являются рациональные функции одной переменной, которые в свою очередь представляются стандартным образом. Если $t = 0$, то оператор может быть также представлен только одной явной матрицей A.

Процедура `IsUnimodular` возвращает `true` или `false` в качестве результата проверки унимодулярности данного на вход оператора. Если указан необязательный входной параметр процедуры — имя переменной, то также вычисляется соответствующий обратный оператор в случае его существования, и вычисленный обратный оператор присваивается этой переменной. Обратный оператор также представляется в виде списка из его явной матрицы и его ведущего и трейлингового порядков. Если необязательное имя переменной не задано, то процедура использует алгоритм Unimodularity Testing, иначе — алгоритм Inverse Operator (см. п. 4). При этом процедура имеет еще один необязательный параметр `method`, который может принимать значения `EG`, `TEG` или `RR` и позволяет указать какой из алгоритмов EG, $\triangle$EG, RR, соответственно, использовать в качестве базового (если этот параметр не указан, то используется EG). В первой версии реализации, представленной в [2], в качестве базового алгоритма использовался только EG.

Пример 1. Рассмотрим применение процедуры `IsUnimodular` к оператору

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

Явная матрица оператора равна

$$\begin{pmatrix} 0 & -\frac{1}{x}\ 1 & 0 \\ 0 & -\frac{x}{2}\ \frac{x^2}{2} & 1 \end{pmatrix},$$

с $l = 1$ и $t = 0$. Процедура вызывается дважды для каждого из методов: первый раз только для проверки унимодулярности, а во второй раз и для построения обратного оператора. Дополнительно выведено и время вычислений:

```
> L := Matrix([[0, -1/x, 1, 0],
               [0, -x/2, x^2/2, 1]]);
```

$$L := \begin{bmatrix} 0 & -\frac{1}{x} & 1 & 0 \\ 0 & -\frac{x}{2} & \frac{x^2}{2} & 1 \end{bmatrix}$$

```
> st:=time():
  IsUnimodular(L, x, 'method'='EG');
  time()-st;
```

$$true$$

$$0.021$$

```
> st:=time():
  IsUnimodular(L, x, 'InvL_EG',
                     'method'='EG');
  time()-st;
```

$$true$$

$$0.026$$

```
> InvL_EG;
```

$$\left[\begin{bmatrix} -\frac{(x+1)^2}{2x} & \frac{1}{x} & 1 & 0 \\ 0 & 0 & -\frac{x^2}{2} & 1 \end{bmatrix}, 1, 0\right]$$

```
> st:=time():
  IsUnimodular(L, x, 'method'='TEG');
  time()-st;
```

$$true$$

$$0.003$$

```
> st:=time():
  IsUnimodular(L, x, 'InvL_TEG',
                     'method'='TEG');
  time()-st;
```

$$true$$

$$0.008$$

```
> InvL_TEG;
```

$$\left[\begin{bmatrix} -\frac{(x+1)^2}{2x} & \frac{1}{x} & 1 & 0 \\ 0 & 0 & -\frac{x^2}{2} & 1 \end{bmatrix}, 1, 0\right]$$

```
> st:=time():
  IsUnimodular(L, x, 'method'='RR');
  time()-st;
```

$$true$$

$$0.032$$

```
> st:=time():
  IsUnimodular(L, x, 'InvL_RR',
                     'method'='RR');
  time()-st;
```

$$true$$

$$0.040$$

```
> InvL_RR;
```

$$\left[\begin{bmatrix} -\frac{(x+1)^2}{2x} & \frac{1}{x} & 1 & 0 \\ & & & \\ 0 & 0 & -\frac{x^2}{2} & 1 \end{bmatrix}, 1, 0\right]$$

Таким образом

$$L^{-1} = \begin{pmatrix} 1 - \frac{(x+1)^2}{2x}\sigma & \frac{1}{x}\sigma \\ & \\ -\frac{x^2}{2} & 1 \end{pmatrix}.$$

□

Пример 2. Оператор

$$M = \begin{pmatrix} I_n & M_1 & 0_n \\ 0_n & I_n & M_2 \\ 0_n & 0_n & I_n \end{pmatrix}, \qquad (6)$$

где 0_n — нулевая матрица размера $n \times n$, $M_1, M_2 \in \mathrm{Mat}_n(\mathbb{K}[\sigma, \sigma^{-1}])$ — произвольные операторы, является унимодулярным для любых M_1 and M_2. Обратный оператор равен

$$M^{-1} = \begin{pmatrix} I_n & -M_1 & M_1M_2 \\ 0_n & I_n & -M_2 \\ 0_n & 0_n & I_n \end{pmatrix}.$$

Выполнена серия экспериментов. Для каждого из них генерировалась пара из разреженных на 50% операторов M_1 и M_2 с элементами в виде случайных полиномов степени не больше 2. Затем для (6) находился обратный оператор. Операторы M_1 и M_2 имели одинаковое число строк, оператор M_1 имел порядок 2, а порядок оператора M_2 принимал значения $d = 3, 5, 7, 9, 11$ (тот же порядок имел и оператор M). Число строк M_1 и M_2 равнялось $n = 2, 3, 4$ (соответственно, число строк M равнялось $k = 4, 6, 8$). Обратный оператор M в каждом эксперименте вычислялся всеми тремя методами. Полученные результаты представлены в таб. 1. Использование метода, основанного на RR, в ряде экспериментов не дало возможности получить результат — вычисления были принудительно остановлены, когда был исчерпан доступный объем физической памяти используемого компьютера. Это помечено в таблице знаком $>$ с указанием времени до принудительной остановки вычислений.

□

Таблица 1.

		d=3	d=7	d=11	d=15
k=6	EG	0.172	0.297	0.500	0.672
	TEG	0.125	0.375	1.032	2.563
	RR	0.937	3.844	5.844	11.500
k=9	EG	0.610	2.562	6.813	21.422
	TEG	0.985	4.594	11.859	26.469
	RR	4.625	82.953	>5100	>6000
k=12	EG	1.125	5.319	31.953	17.422
	TEG	3.219	15.735	36.312	59.312
	RR	18.250	170.078	>8300	>19100
k=15	EG	3.657	49.219	215.329	908.984
	TEG	8.047	41.516	90.531	213.032
	RR	1031.422	>20200	>6100	>21100

6. АНАЛИЗ РАСХОЖДЕНИЙ

Из таб. 1 видно, что результаты в целом соответствуют теоретическим оценкам сложности. При этом относительный рост времени вычислений с ростом числа строк и порядка оператора происходит заметно быстрее, чем это можно было ожидать, исходя из этих теоретических оценок. Чтобы понять причину этих расхождений нами был реализован режим работы процедуры `IsUnimodular`, в котором дополнительно выводятся характеристические параметры промежуточных результатов вычислений. Эти параметры выводятся после каждого появления нулевой строки в выявляющей матрице (после выполнения редукции в алгоритмах EG и RR или после выполнения одной или нескольких замен строк в алгоритме $\triangle$EG). В качестве таких параметров используются параметры строки оператора, в которой появилась нулевая строка в выявляющей матрице, а именно объем данных в представлении этой строки: используется стандартная `Maple`-функция `length`, а также максимум сумм степеней числителя и знаменателя коэффициентов в этой строке оператора. Эти дополнительные данные показали, что в ходе промежуточных вычислений коэффициенты оператора значительно "распухают". Несмотря на то, что в эксперименте суммы степеней числителей и знаменателей коэффициентов в исходном операторе M не превышает 2, а в обратном к нему не превышает 4, в промежуточных вычислениях эта величина достигала сотен и даже тысяч. Столь же значительно рос и объем данных.

В таб. 2 представлены сводные данные по тем же экспериментам, которые представлены в 1. Для каждого эксперимента представлено пять параметров:

- максимальный среди всех промежуточных строк объем используемых данных,
- максимальная среди всех промежуточных строк величина максимума суммы степеней числителя и знаменателя,
- число промежуточных строк,
- средний по всем промежуточным строкам объем

используемых данных,

- средняя по всем промежуточным строкам величина максимума суммы степеней числителя и знаменателя.

Для тех экспериментов, которые были принудительно остановлены, указываются параметры, вычисленные для промежуточных результатов, полученных до остановки вычислений.

Представленный в таб. 1 значительный рост времени вычислений в экспериментах с ростом числа строк и порядка оператора согласуется с "распуханием" коэффициентов в промежуточных вычислениях, представленных в таб. 2 характеристическими параметрами. В результате такого "распухания" фактические вычислительные затраты на выполнение одной арифметической операции, как и операции сдвига, начинают существенно расти, и этот рост оказывается более существенным в сравнении с ростом числа самих таких операции, отраженных в сложности алгоритмов. Для разных методов, это "распухание" происходит с разной скоростью. В наших экспериментах с этой точки зрения особенно плохо проявил себя алгоритм RR. Это согласуется с результатами сравнения аналогичных алгоритмов в дифференциальном случае, представленными в [1]. Отметим также, что в условиях, когда "распухание" в разных алгоритмах имеет схожие характеристики, то есть затраты на выполнения одной операции в каждом алгоритме сопоставимы, то на первый план снова выходит число этих операций, и относительное время вычислений больше соответствует теоретическим оценкам сложности.

И отметим еще одну особенность алгоритмов EG и RR. Для выполнения редукции в этих алгоритмах используется поиск линейных зависимостей в выявляющих матрицах. В реализации для этого используется процедура `NullSpace` пакета `LinearAlgebra` системы `Maple`. Эта процедура может найти более чем одну линейную зависимость, что дает возможность различного продолжения вычислений, в зависимости от выбора для исключений той или иной из найденных линейных зависимостей. В нашей реализации выбиралась первая из найденных линейных зависимостей после их сортировки с помощью процедуры `ComplexitySort` пакета `SolveTools`. В таб. 3, 4 представлены результаты вычислений для части тех же тестовых систем, что и в таб. 1, 2, но с использованием выбора не первой, а последней линейной зависимости после сортировки. Представленные результаты показывают, что порядок исключений также существенно влияет на "распухание" коэффициентов в промежуточных вычислениях, а значит и на общее время вычислений. Этот фактор также не принимается во внимание в оценках сложности, однако с практической точки зрения является весьма существенным.

Исходя из проведенных экспериментов, а также из ранее проведенных экспериментов для дифференциального случая ([1]), с практической точки зрения разумно использовать версию алгоритмов Unimodularity Testing и Inverse Operator, основанных на алгоритме EG. Именно поэтому такой вариант реализации был представлен в [2] и используется по умолчанию (если не указан параметр `method`) в реализации в этой работе. Эти эксперименты также показывают, что с практической точки зрения полезно иметь реализации различных алгоритмов. Для конкретного примера более эффективным может оказаться алгоритм, уступающий другим в сложностном отношении или времени счета в других примерах. Поэтому в нашей реализации оставлена возможность использования и других алгоритмов по выбору пользователя.

Авторы благодарны А.А.Рябенко за замечания по первому варианту статьи и полезные советы.

СПИСОК ЛИТЕРАТУРЫ

1. С.А.Абрамов, А.А.Рябенко, Д.Е.Хмельнов. Выявляющие матрицы линейных дифференциальных систем произвольного порядка. *Программирование*, No 2, С. 7-16 (2017).

2. S.A. Abramov, D.E. Khmelnov. On unimodular matrices of difference operators. *CASC 2018 Proceedings*, vol. 11077, pp.18–31 (2018).

3. Abramov, S. EG–Eliminations. *J. Difference Equations Appl.* 5, 393–433 (1999)

4. Abramov, S., Bronstein, M. Linear algebra for skew-polynomial matrices. Rapport de Recherche INRIA RR-4420, March 2002 `http://www.inria.fr/RRRT/RR-4420.html`

5. Abramov, S.: On the Differential and Full Algebraic Complexities of Operator Matrices Transformations. *CASC 2016, Proceedings, LNCS*, vol. 9890, pp. 1–14 (2016).

6. Abramov, S., Barkatou, M. On the dimension of solution spaces of full rank linear differential systems. *CASC 2013, Proceedings, LNCS*, vol. 8136, pp. 1–9 (2013).

7. Abramov, S., Barkatou, M. On solution spaces of products of linear differential or difference operators. *ACM Comm. in Computer Algebra*, 4, 155–165 (2014).

8. Кац В.Г., Чен П. Квантовый анализ. М: МЦНМО, 2005.

9. 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., V. 66 (1986).

10. Абрамов, С. А. Обратные линейные разностные операторы. *Журнал вычислительной математики и математической физики*, том 57, № 12, с. 1933—1945 (2017).

11. Abramov, S., Bronstein, M. On solutions of linear functional systems. *ISSAC'2001 Proceedings*, pp. 1–6 (2001)

12. Knuth, D.E. Big omicron and big omega and big theta. *ACM SIGACT News*, 8(2), 18–23 (1976)

13. С.А.Абрамов. Лекции о сложности алгоритмов. Издание второе, переработанное. М: МЦНМО, 2012.

14. 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.*, V. 49. No. 1. P. 45–64 (2013).

15. Beckermann, B., Cheng, H., Labahn, G. Fraction-free row reduction of matrices of Ore polynomials. *J. Symb. Comput.*, 41, 513–543 (2006).

16. Mulders T., Storjohann A. On lattice reduction for polynomial matrices. *J. Symb. Comput.*, V. 37, No. 4. P. 485–510 (2004).

17. Maple online help: https://www.maplesoft.com/support/help/

18. С.А.Абрамов, А.А.Рябенко, Д.Е.Хмельнов. Лорановы, рациональные и гипергеометрические решения линейных q-разностных систем произвольного порядка с полиномиальными коэффициентами. *Программирование*, No 2, с. 60–73 (2018).

Таблица 2.

		d=3	d=7	d=11	d=15
k=6	EG	88	151	227	294
		2	2	2	2
		11	17	25	30
		63	105	154	201
		1	1	1	2
	TEG	138	1169	11188	52729
		3	10	28	51
		8	14	23	31
		85	333	1870	11334
		2	4	9	18
	RR	843	10488	18719	42013
		17	39	46	58
		9	17	23	31
		281	2162	2757	9759
		6	15	12	22
k=9	EG	1779	36697	174632	556987
		20	145	82	122
		21	33	45	57
		291	6134	32615	116131
		4	26	28	46
	TEG	3411	29855	161977	411703
		31	54	94	129
		15	27	39	51
		1132	11140	51255	114004
		13	30	46	68
	RR	3702	363550	>80150107	>186956920
		27	245	>2532	>2870
		15	27	>19	>14
		1379	119074	12653623	28598185
		16	119	601	693
k=12	EG	1460	64793	458748	332049
		8	56	113	88
		27	43	59	74
		261	11011	84783	50312
		2	15	38	24
	TEG	9414	63490	202193	340476
		54	96	96	117
		20	36	52	67
		2872	27979	88520	121198
		19	44	64	69
	RR	26360	460957	>190475287	>168181419
		58	190	>2902	>2303
		20	36	>20	>33
		6169	95011	23245248	24965627
		29	74	611	539
k=15	EG	24060	346779	1383502	3938486
		39	110	170	236
		34	54	74	94
		3023	58838	259553	831962
		10	35	62	93
	TEG	58343	131895	400249	2066923
		110	114	112	222
		24	44	64	84
		11782	60119	115177	600333
		37	66	74	119
	RR	981268	>87692555	>307185043	>472458234
		692	>2130	>2944	>2800
		24	>16	>11	>15
		137728	17253891	39424269	59156406
		157	643	599	595

Таблица 3.

		d=3	d=7
k=6	EG	0.187	0.359
	RR	0.500	2.406
k=9	EG	0.829	20.062
	RR	4.047	29.688
k=12	EG	19.906	16645.953
	RR	29.547	14979.109

Таблица 4.

		d=3	d=7
k=6	EG	88	460
		2	4
		11	19
		63	155
		1	2
	RR	87	438
		2	4
		6	14
		65	172
		1	2
k=9	EG	4178	186239
		23	174
		21	28
		570	19817
		5	39
	RR	3717	165811
		23	174
		15	20
		695	24107
		7	54
k=12	EG	186239	18578353
		174	1096
		28	44
		19817	2422729
		39	233
	RR	165811	15621336
		174	1096
		20	36
		24107	2512118
		54	284