ПРОГРАММИРОВАНИЕ, 2017, № 2, с. 7–16

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

ВЫЯВЛЯЮЩИЕ МАТРИЦЫ ЛИНЕЙНЫХ ДИФФЕРЕНЦИАЛЬНЫХ СИСТЕМ ПРОИЗВОЛЬНОГО ПОРЯДКА*

© 2017 г. С.А. Абрамов, А.А. Рябенко, Д.Е. Хмельнов

Вычислительный центр им. А.А.Дородницына ФИЦ ИУ РАН

119333 Москва, ул. Вавилова, 40

E-mail: sergeyabramov@mail.ru, anna.ryabenko@gmail.com, dennis_khmelnov@mail.ru

Поступила в редакцию 30.09.2016

Известно, что если ведущая матрица линейной дифференциальной системы невырождена, то ее определитель несет полезную информацию о решениях системы. Может представлять интерес и фронтальная матрица. Но каждая из этих матриц (будем называть их выявляющими) может оказаться вырожденной. Мы даем краткий обзор алгоритмов преобразования системы полного ранга к системе с невырожденной выявляющей матрицей запрашиваемого типа. Эти же преобразования позволяют проверять полноту ранга исходной системы. Предлагается Maple-реализация этих алгоритмов (пакет `EGRR`) и приводятся результаты сопоставления оценок их сложности с фактическим временем работы для ряда примеров.

1. ВВЕДЕНИЕ

Системы линейных обыкновенных дифференциальных уравнений возникают во многих областях математики. Одним из направлений компьютерной алгебры является разработка новых алгоритмов поиска решений дифференциальных уравнений и их систем, а также алгоритмов, которые могут использоваться при таком поиске в качестве вспомогательных или как составная часть известных алгоритмов поиска решений. Ведущая и фронтальная матрицы системы (эти понятия вводятся и обсуждаются в п.п. 2.1, 2.2 раздела 2) в случае их невырожденности позволяют, например, определить возможные особые точки решений системы: множество корней определителя любой из этих матриц включает в себя все особые точки. Но определитель каждой из этих матриц (мы называем эти матрицы *выявляющими*) может оказаться тождественно нулевым.

Мы даем краткий обзор алгоритмов преобразования системы полного ранга к системе с невырожденной выявляющей матрицей запрашиваемого типа. Различные алгоритмы предварительно сравниваются между собой с помощью анализа их сложности по числу операций в худшем случае. При этом к числу учитываемых операций отнесена и операция дифференцирования (в [1] показано, что алгоритмы, имеющие одинаковую сложность по числу арифметических операций, могут иметь разные сложности по числу всех используемых операций, включая дифференцирование).

Обсуждается реализация рассматриваемых алгоритмов и приводятся результаты экспериментального сравнения. При этом выясняется, что для каждого из алгоритмов можно найти случаи, когда именно этот алгоритм работает быстрее других (это не противоречит результатам анализа сложности, так как не все случаи являются “худшими” в том смысле, в котором эти случаи понимаются при исследовании сложности и, помимо этого, обсуждаемая сложность по числу операций не учитывает размеров операндов; сравнение алгоритмов на основе обсуждаемой сложности носит лишь предварительный или ориентировочный характер). Пользователь может выбрать на основе

*Частичная поддержка РФФИ, грант 16-01-00174-а.

своих критериев тот или иной из алгоритмов, возможность выбора оказывается актуальной, когда данная система не поддается быстрому решению, и, вообще, не вполне ясно, как к этой системе подступиться.

2. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

2.1. *Системы и операторы*

Если R — некоторое кольцо (в частности, поле), то $\mathrm{Mat}_m(R)$ обозначает кольцо $m\times m$-матриц с элементами из R. Обозначение M^T используется для матрицы, транспонированной к M, обозначение $M_{i,*}$, $1 \leqslant i \leqslant m$, — для $1 \times m$-матрицы, которая совпадает с i-й строкой $m \times m$-матрицы M.

Пусть $\mathbb{K}$ — дифференциальное поле характеристики 0 с производной $\partial = '$. Мы будем рассматривать системы вида

$$A_r\partial^r y + A_{r-1}\partial^{r-1}y + \cdots + A_0 y = 0, \qquad (1)$$

где $y = (y_1, y_2, \ldots, y_m)^T$ — вектор неизвестных функций от x. Относительно

$$A_0, A_1, \ldots, A_r$$

предполагается, что $A_i \in \mathrm{Mat}_m(\mathbb{K})$, $i = 0, 1, \ldots, r$, при этом A_r (*ведущая* матрица системы) является ненулевой.

Элементы матриц A_i будем называть *коэффициентами системы* (1).

Система (1) может быть записана как $L(y) = 0$ с оператором L

$$A_r\partial^r + A_{r-1}\partial^{r-1} + \cdots + A_0, \qquad (2)$$

число r является *порядком* оператора L (пишем $r = \operatorname{ord} L$).

При $m = 1$ оператор (2) становится *скалярным* оператором, имеющим вид полинома от ∂. Такие операторы складываются и вычитаются как обычные полиномы, при умножении используется переместительное правило $\partial a = a\partial + a'$ (или, то же самое, $\partial a = a\partial + \partial(a)$). Произведение — это, фактически, композиция операторов. Для обозначения кольца этих скалярных операторов привлекается стандартное обозначение кольца полиномов от ∂ над $\mathbb{K}$, т.е. $\mathbb{K}[\partial]$. Понятие порядка естественно определяется и для скалярных операторов, т.е. элементов кольца $\mathbb{K}[\partial]$.

Оператор (2) может быть представлен единой операторной матрицей, принадлежащей $\mathrm{Mat}_m(\mathbb{K}[\partial])$:

$$\begin{pmatrix} L_{11} & \ldots & L_{1m} \\ \ldots & \ldots & \ldots \\ L_{m1} & \ldots & L_{mm} \end{pmatrix}, \qquad (3)$$

$L_{ij} \in \mathbb{K}[\partial]$, $i, j = 1, \ldots, m$, при этом $\max_{i,j} \operatorname{ord} L_{ij} = r$. Порядком i-й строки матрицы (оператора) (3) называется наибольший из порядков скалярных операторов, составляющих эту строку, т.е.

$$\operatorname{ord} L_{i,*} = \max_{j=1}^{m} \operatorname{ord} L_{i,j}.$$

Будем говорить, что строки с номерами $i_1, \ldots, i_s$, $s \leqslant m$, оператора (3) *линейно независимы* над $\mathbb{K}[\partial]$, если из того, что их линейная комбинация с левыми множителями $f_1, \ldots, f_s \in \mathbb{K}[\partial]$ равна нулю, т.е. из того, что $f_1 L_{i_1,*} + \cdots + f_s L_{i_s,*} = 0$, следует, что $f_1 = \cdots = f_s = 0$.

Система может быть задана в операторной форме с одним из двух представлений операторов. В дальнейшем мы будем выбирать форму представления исходя из соображений удобства.

Матрица A_r является ведущей матрицей системы $L(y) = 0$ и оператора L независимо от формы представления системы и оператора.

Если все строки оператора (3) линейно независимы над $\mathbb{K}[\partial]$, то мы называем уравнения соответствующей системы независимыми над $\mathbb{K}[\partial]$. В этом случае оператор $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ имеет *полный ранг*, как и система $L(y) = 0$. Мы будем также называть их оператором и системой *полного ранга*. Именно такие операторы и системы будут для нас представлять основной интерес.

2.2. *Размерность пространства решений*

Пусть поле констант $Const(\mathbb{K}) = \{c \in \mathbb{K} \mid \partial c = 0\}$ поля $\mathbb{K}$ алгебраически замкнуто. Обозначим через Λ фиксированное *универсальное расширение Пикара–Вессио* для $\mathbb{K}$ (см. [2, Sect. 3.2]). Это будет некоторое дифференциальное расширение Λ поля $\mathbb{K}$, для которого $Const(\Lambda) = Const(\mathbb{K})$ и такое, что

любая дифференциальная система $\partial y = Ay$, $A \in \mathrm{Mat}_m(\mathbb{K}[\partial])$, имеет в Λ^m пространство решений размерности m над полем констант. Для произвольного оператора L вида (3) обозначим через V_L линейное над $Const(\Lambda)$ пространство решений L (т.е. решений системы уравнений $L(y) = 0$), принадлежащих Λ^m. Его размерность будет обозначаться через $\dim V_L$.

Предположим, что $Const(\mathbb{K})$ не является алгебраически замкнутым. Для каждого дифференциального поля $\mathbb{K}$ характеристики 0 существует дифференциальное расширение, имеющее алгебраически замкнутое поле констант: это, например, алгебраическое замыкание $\bar{\mathbb{K}}$ с производной, полученной естественным расширением производной в поле $\mathbb{K}$. В этом случае $Const(\bar{\mathbb{K}}) = \overline{Const(\mathbb{K})}$ (см. [2, Exercises 1.5, 2:(c),(d)], [3, Sect. 3]). Тогда V_L является линейным над $Const(\bar{\mathbb{K}})$ пространством решений L, компоненты которых принадлежат универсальному дифференциальному расширению поля $\bar{\mathbb{K}}$.

Пусть оператор L полного ранга имеет вид (2). Если $1 \leqslant i \leqslant m$, то определим $\alpha_i(L)$ как наибольшее целое k, $1 \leqslant k \leqslant r$, такое, что $(A_k)_{i,*}$ есть ненулевая строка. Таким образом, $\alpha_i(L) = \mathrm{ord}\, L_{i,*}$.

Матрица $F \in \mathrm{Mat}_m(\mathbb{K})$ такая, что $F_{i,*} = (A_{\alpha_i(L)})_{i,*}$, $i = 1, \ldots, m$, называется *фронтальной матрицей* оператора L.

Группу унимодулярных операторов из $\mathrm{Mat}_m(\mathbb{K}[\partial])$ будем обозначать посредством Υ_m.

Под дифференцированием строки $L_{i,*}$ оператора (3) (то же самое — под применением ∂ к этой строке) будем понимать замену $(L_{i1}, \ldots, L_{i,m})$ строкой $(\partial L_{i1}, \ldots, \partial L_{i,m})$, где ∂L_{ij} — это композиция (произведение в $\mathbb{K}[\partial]$) скалярных операторов ∂ и L_{ij}.

Сформулируем теорему из [1], являющуюся следствием утверждений, доказанных в [4, 5] (часть (iii) этой теоремы может быть также доказана с использованием [6, Thm III]).

Теорема 1. *Пусть $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ имеет полный ранг. Тогда*

(i) Если L' служит результатом дифференцирования какой-то строки L, то $\dim V_{L'} = \dim V_L + 1$.

(ii) Если фронтальная матрица для $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ невырождена, то

$$\dim V_L = \sum_{i=1}^{m} \alpha_i(L).$$

(iii) $L \in \Upsilon_m \iff V_L = 0$.

В дальнейшем мы предполагаем, что поле $\mathbb{K}$ конструктивно, в частности, что существует процедура проверки равенства нулю заданного элемента поля.

2.3. *Алгоритм EG (EG-исключения)*

Для заданного оператора $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ полного ранга алгоритм EG ([4, 7, 8, 9]) строит некоторую *охватывающую* операторную матрицу $\widehat{L} \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ такую, что

- $\mathrm{ord}\, \widehat{L} = \mathrm{ord}\, L$,
- $\widehat{L}$ имеет невырожденную ведущую матрицу,
- $\widehat{L} = QL$ при $Q \in \mathrm{Mat}_m(\mathbb{K})$, таким образом, $V_L \subseteq V_{\widehat{L}}$.

Если ранг заданного оператора L не полон, то алгоритм сообщает об этом.

Алгоритм строит $\widehat{L}$ на месте самого оператора L, т.е. L изменяется шаг за шагом, постепенно преобразуясь в $\widehat{L}$:

> Проверить, являются ли строки ведущей матрицы оператора L линейно зависимыми над $\mathbb{K}$. Если нет, то L не изменяется и алгоритм останавливается. В противном случае следует серия шагов, каждый из которых состоит из двух этапов.
>
> *Этап редукции:*
> Вычислить коэффициенты $p_1, \ldots, p_m \in \mathbb{K}$ зависимости строк ведущей матрицы. Взять какое-нибудь i, $1 \leqslant i \leqslant m$, для которого $p_i \neq 0$.
> Заменить строку $L_{i,*}$ в (3) на
>
> $$\sum_{k=1}^{m} p_k L_{k,*}. \qquad (4)$$
>
> В ведущей матрице оператора L при этом i-я строка становится нулевой.

Этап дифференциального сдвига: К i-й строке (значение i выбрано на этапе редукции) применить $\partial^{r-\alpha_i}$, в результате $\alpha_i = r$.

Если в некоторый момент в L появляется нулевая строка или число шагов "редукция + дифференциальный сдвиг" оказывается равным $mr+1$, то это означает, что исходный оператор L не имел полного ранга. В остальных случаях получаем оператор с невырожденной ведущей матрицей и применение алгоритма потребует не более mr шагов "редукция + дифференциальный сдвиг".

Названная оценка числа шагов обосновывается тем, что на каждом шаге "редукция + дифференциальный сдвиг" этап редукции не изменяет пространства решений, а этап дифференциального сдвига увеличивает по теореме 1(i) размерность пространства решений за счет дифференцирований. При этом ни на одном шаге размерность пространства решений не может превышать mr. Таким образом, и общее число шагов алгоритма EG не может превышать mr. Каждый из этапов редукции и дифференциального сдвига эквивалентен умножению слева исходного оператора на некоторый оператор, поэтому в итоге получается оператор вида QL, где $Q \in \mathrm{Mat}_m(\mathbb{K}[\partial])$. (При желании или необходимости построение матрицы Q может быть включено в алгоритм EG.)

В [1, Prop. 1] проведено исследование сложности алгоритма EG и некоторых других алгоритмов, о которых речь пойдет в пп. 2.4, 2.5. Сложность понимается как число операций в поле $\mathbb{K}$ в худшем случае при фиксированных m и r. К учитываемым операциям относятся как арифметические операции поля, так и операция дифференцирования. Для такой сложности $T_{\mathrm{EG}}(m,r)$ алгоритма EG показано, что

$$T_{\mathrm{EG}}(m,r) = \Theta(m^{\omega+1}r + m^3r^2), \qquad (5)$$

где ω — показатель матричного умножения, $2 < \omega \leqslant 3$.

2.4. *Алгоритм RR*

В основе лежит алгоритм FF ([10]). Упрощенный вариант алгоритма FF в [11] назван RowReduction; для краткости мы в дальнейшем будем использовать сокращение RR.

Для данного оператора $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ полного ранга алгоритм RR строит такой оператор $\breve{L} \in \mathrm{Mat}_m(\mathbb{K}[\partial])$, что

- $\operatorname{ord}\breve{L} \leqslant \operatorname{ord} L$,
- фронтальная матрица оператора $\breve{L}$ невырождена,
- $\breve{L} = UL$ для некоторого $U \in \Upsilon_m$, и, как следствие, $V_L = V_{\breve{L}}$.

Опишем коротко этот алгоритм, сосредоточившись на построении $\breve{L}$. Оператор $\breve{L}$ строится на месте оператора L, т.е. L изменяется шаг за шагом, постепенно преобразуясь в $\breve{L}$:

> Проверить, являются ли строки фронтальной матрицы оператора L линейно зависимыми над $\mathbb{K}$. Если нет, то $\breve{L} = L$, и алгоритм останавливается. В противном случае пусть $p_1, \ldots, p_m \in \mathbb{K}$ — коэффициенты зависимости. Из тех строк (3), которым соответствуют ненулевые коэффициенты, выбрать строку, имеющую наибольший порядок (если есть несколько строк с этим значением порядка, то взять любую из них). Пусть это будет i-я строка, т.е. $L_{i,*}$. Строка $L_{i,*}$ оператора L заменяется на
>
> $$\sum_{j=1}^{m} p_j \partial^{\alpha_i - \alpha_j} L_{j,*}. \qquad (6)$$
>
> После конечного числа шагов получаем оператор с невырожденной фронтальной матрицей.
>
> Если в некоторый момент выполнения алгоритма в L возникает нулевая строка, то ранг L не полон.

Завершимость гарантируется тем, что на каждом шаге убывает сумма порядков строк оператора L.

В [1, Prop. 2] показано, что

$$T_{\mathrm{RR}}(m,r) = \Theta(m^{\omega+1}r + m^3r^3). \qquad (7)$$

Обратим внимание, что максимальный показатель степени, с которым r входит в (5), меньше, чем соответствующий показатель в (7). Но зато $V_L = V_{\breve{L}}$, тогда как для $\widehat{L}$ возможно, что $\dim V_L < \dim V_{\widehat{L}}$.

2.5. *Алгоритмы $\triangle EG$ и $\triangle RR$*

Пусть i-я строка фронтальной матрицы оператора $L \in \mathrm{Mat}_m(\mathbb{K}[\partial])$ имеет вид

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

$1 \leqslant k \leqslant m$, $a \neq 0$. Тогда число k будем называть *пин-индексом* i-й строки L. Если i-я строка оператора L нулевая, то считаем, что ее пин-индекс равен $-\infty$. Но мы рассмотрим случай L полного ранга.

Если строки L имеют попарно различные пин-индексы, то фронтальная матрица невырождена: с точностью до порядка строк это будет треугольная матрица с ненулевыми диагональными элементами. Пусть строки $r_1 = L_{i,*}$ и $r_2 = L_{j,*}$ имеют одинаковые пин-индексы равные k, пусть $\operatorname{ord} L_{i,*} = d_1$, $\operatorname{ord} L_{j,*} = d_2$ и при этом $d_1 \leqslant d_2$. Возьмем $v \in \mathbb{K}$ такое, что строка

$$r_2 - v\partial^{\,d_2-d_1} r_1 \tag{8}$$

имеет либо пин-индекс, больший, чем k, либо же порядок, меньший, чем d_2. В операторе L строка r_2, т.е. $L_{j,*}$, заменяется строкой (8). Если L имеет полный ранг, то фронтальная матрица после нескольких шагов описанного вида становится треугольной. Это может быть использовано[1] вместо поиска линейной зависимости строк фронтальной матрицы (для EG ведущая и фронтальные матрицы в соответствующие моменты совпадают и $d_2 - d_1 = 0$ в (8)).

Получаем новые варианты алгоритмов EG и RR, которые мы назовем $\triangle$EG и соответственно $\triangle$RR. В [1, Prop. 3] было показано, что

$$T_{\triangle\mathrm{EG}}(m,r) = \Theta(m^3r^2) \tag{9}$$

и

$$T_{\triangle\mathrm{RR}}(m,r) = \Theta(m^3r^3). \tag{10}$$

Близкий подход, но без оценок (9), (10), изложен в [10, Sect. 9.1].

[1] Для разностного случая это уже использовалось в [12] в первой версии алгоритма EG. В дискуссии, касающейся дифференциального случая, А.Шторйоханн привлек внимание первого из авторов к тому факту, что сложность этого подхода меньше, чем подхода, связанного с решением линейных алгебраических систем (см. также [13]).

2.6. *Если продифференцированные строки сохраняются*

Можно сохранять все результаты дифференцирований строк. Для этого случая в [1] получены некоторые верхние оценки общего числа дифференцирований.

Предложение 1. *[1, Prop. 6] Число дифференцирований без повторений (когда результат каждого дифференцирования сохраняется) выполняемых алгоритмами RR и $\triangle RR$ есть $O(mr^2)$ и соответственно $O(m^2r^2)$ в худшем случае; как следствие, число дифференцирований элементов поля $\mathbb{K}$ есть $O(m^2r^3)$ и соответственно $O(m^3r^3)$.*

Однако оценки $O(m^2r^3)$ и $O(m^3r^3)$ для числа дифференцирований не позволяют понизить показатель степени для r в (7), (10). Основываясь на предложении 1, мы не можем сделать заключение, что хранение результатов всех дифференцирований существенно снижает сложности алгоритмов RR и $\triangle$RR. Но пространственная сложность заметно возрастает, когда мы сохраняем все результаты дифференцирований.

При этом у нас нет доводов в пользу того, что, например, верхняя оценка $O(mr^2)$ числа дифференцирований строк алгоритмом RR является в каком-либо смысле точной. Интересный вопрос, ответ на него не ясен, — справедлива ли оценка $O(mr)$?

При всем этом пространственная сложность (затраты дополнительной памяти в худшем случае) заметно возрастает, когда мы сохраняем все результаты дифференцирований.

3. РЕАЛИЗАЦИЯ

Рассмотренные алгоритмы реализованы в системе компьютерной алгебры `Maple` ([14]) в виде процедур нового пакета `EGRR`[2]. Реализация выполнена для систем, коэффициенты которых — рациональные функции одной переменной над полем рациональных чисел.

Пакет предоставляет процедуры

- `EG`: реализует алгоритм EG,

[2] Пакет и сессия `Maple` с примерами использования описываемых процедур доступны по адресу http://www.ccas.ru/ca/egrr

- `RR`: реализует алгоритм RR,
- `TriangleEG`: реализует алгоритм $\triangle$EG,
- `TriangleRR`: реализует алгоритм $\triangle$RR.

3.1. *Входные параметры и возвращаемые значения процедур*

Дифференциальная система (1) задается во входных параметрах процедур пакета `EGRR` в виде *явной матрицы системы* $(A_r|A_{r-1}|\ldots|A_0)$, имеющей размер $m \times m(r+1)$. В свою очередь, эта матрица системы задается с помощью стандартного объекта `Matrix`. Элементы явной матрицы — рациональные функции одной переменной, они также задаются стандартным в `Maple` образом.

Каждая из процедур пакета `EGRR` принимает три входных параметра:

- `M` – явная матрица исходной системы;
- `d` – число *блоков* явной матрицы, $d = r + 1$;
- `x` – независимая переменная системы.

В качестве результата возвращаются

- `res` – система, после выполненных в ней преобразований с помощью алгоритма, реализованного в процедуре;
- `full_rank` – `true` в случае, если в ходе работы алгоритма было определено, что система имеет полный ранг, иначе — `false`.

Применение процедур пакета к системе полного ранга приведено на рис. 1.

3.2. *Определение зависимости строк матрицы*

Алгоритмы EG и RR используют в своей работе алгоритм определения линейной зависимости строк матрицы. Для решения этой задачи использована процедура `NullSpace` стандартного пакета `LinearAlgebra` системы `Maple`.

В статье [8], где была впервые высказана идея поиска линейных зависимостей строк ведущей матрицы без попутного преобразования всего оператора L, говорилось, что поиск этих зависимостей может выполняться процедурами пакета `LinearAlgebra`, среди которых имеется немало высокоэффективных. Их привлечение может дать экономию времени при использовании той версии алгоритма EG, которая была представлена в п. 2.3 (см. данное там описание этапа редукции). В [8] подчеркивалось, что коэффициенты линейной зависимости являются решением некоторой системы линейных однородных алгебраических уравнений, и все элементы любого базиса пространства решений этой системы после несложных преобразований могут быть использованы на последовательных этапах редукции в алгоритме EG (аналогичная возможность для алгоритма RR отмечена в [5, Sect. 4.4]).

4. ЭКСПЕРИМЕНТЫ

4.1. *Случайный выбор систем*

Для каждой пары m, r при $m = 4, 8, 10$, $r = 3, 9, 27$ было сгенерировано по 12 систем. Коэффициенты всех систем — случайные полиномы (использовалась стандартная команда Maple `randpoly`, коэффициенты полиномов выбирались из диапазона $[-99, 99]$, а их степени не превышали 8). Системы генерировались разреженными, элемент был ненулевым с вероятностью $\frac{1}{[m/2]+1}$, $\frac{1}{[m/3]+1}$ или $\frac{1}{[m/4]+1}$, где $[...]$ — целая часть числа. Для каждого из вариантов генерировалось 4 системы из 12. В ходе экспериментов фиксировалось время выполнения алгоритма, число выполненных операций дифференцирования элементов матрицы системы, число выполненных арифметических операций.

Результаты экспериментов представлены в табл. 1, содержащей информацию о максимальном времени[3] и максимальном числе операций, затраченных на выполнение алгоритмов.

Приведенные результаты показывают, что изменение максимального числа операций для решения одной системы и худшего (максимального) времени выполнения алгоритма для одной системы при изменении r и m ведут себя по разному — худшее время растет быстрее чем максимальное число операций при росте r и m; при

[3] В секундах. Вычисления проводились в Maple 2016, Ubuntu 8.04.4 LTS, AMD Athlon(tm) 64 Processor 3700+, 3GB RAM.

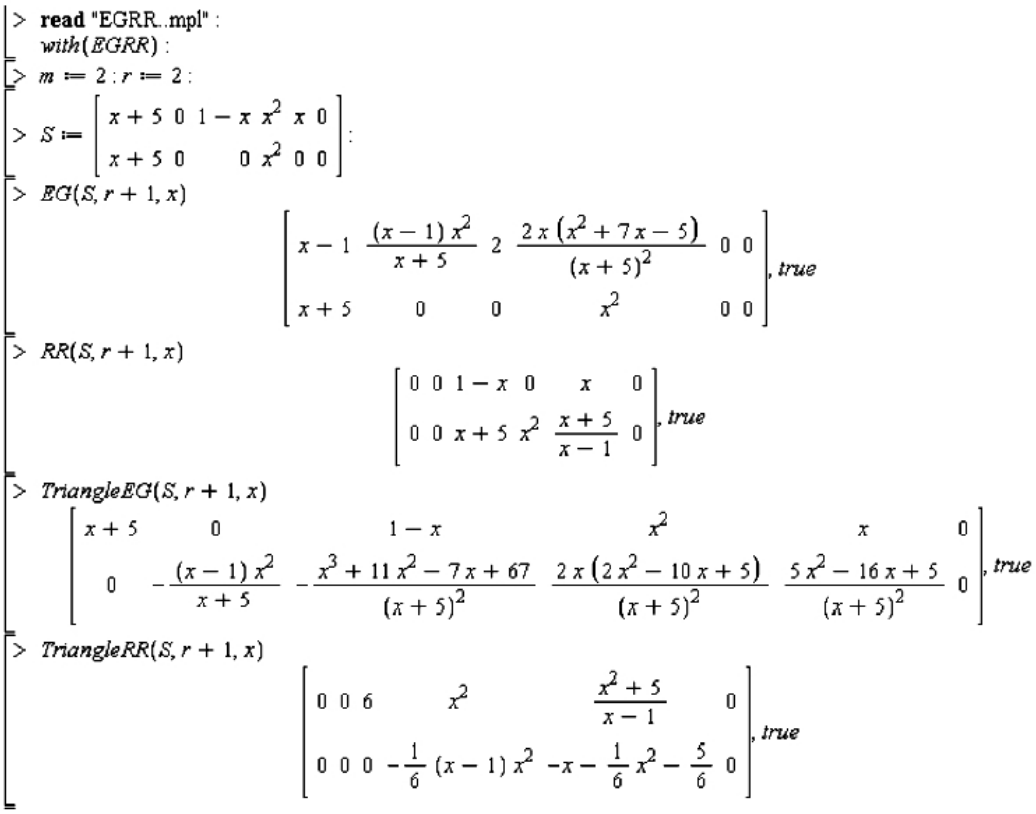

Рис. 1.

этом худшее время не всегда соответствует максимальному числу операций. Это объясняется тем, что время, затрачиваемое на одну операцию не является постоянным. В частности, результаты показывают, что для алгоритмов $\triangle$EG и $\triangle$RR рост затрат на одну операцию с ростом r и m происходит быстрее, чем для EG и RR. В результате, при больших m и r худшее время для $\triangle$EG и $\triangle$RR существенно больше, чем для EG и RR, несмотря на то, что максимальное число операций для $\triangle$EG и $\triangle$RR меньше, чем для EG и RR. Это вызвано "распуханием" элементов явной матрицы при вычислениях, о котором будет сказано подробнее в п. 4.3.

4.2. Запоминание продифференцированных строк

Как отмечалось в п. 2.6, оценки, полученные нами для числа дифференцирований без повторения, не позволяют понизить показатель степени 3 при r в оценках сложности (7), (10), и вопрос о том, можно ли в (7), (10) понизить этот показатель до 2 остается открытым. Эксперименты показывают, что использование опции `remember` для процедуры дифференциального сдвига строки слабо улучшает время работы алгоритмов RR и $\triangle$RR, а иногда и ухудшает его, при том, что число дифференцирований может заметно уменьшиться. В табл. 2 приведены результаты работы RR и $\triangle$RR (время счета и число дифференцирований) над такими случайно сгенерированными системами с числом уравнений $m = 6$, что половина уравнений системы имеет порядок $r = 3, 2, 4, 5, 6, 7, 8, 9$, а другая половина имеет порядок равный 1, и после применения алгоритмов получается система порядка 1. Для таких систем выполняется много повторных дифференцирований строк, но улучшение по времени при их хранении оказалось незначительным.

4.3. Сравнительное рассмотрение EG, RR и $\triangle$EG, $\triangle$RR.

Оценки (5), (7), (9), (10) касаются сложности по числу операций в поле $\mathbb{K}$, они не учитывают размеров операндов. Битовая сложность была бы в этом смысле более информативна, но ее

Таблица 1.

		$r=3$			$r=9$			$r=27$		
		$m=4$	$m=8$	$m=16$	$m=4$	$m=8$	$m=16$	$m=4$	$m=8$	$m=16$
EG	М.В.	0.024	0.294	5.360	0.140	1.733	1.763	0.300	3.590	40.747
	Ч.О.М.В.	608	4920	32992	1624	8592	15664	3252	27600	121776
	М.Ч.О.	608	4920	36640	1624	8592	50896	4712	27600	123856
	В.М.Ч.О.	0.024	0.294	1.574	0.140	1.733	1.743	0.110	3.590	3.800
RR	М.В.	0.034	0.336	5.386	0.150	1.653	2.140	0.424	5.507	57.41
	Ч.О.М.В.	408	4912	21856	1524	7704	40880	3276	25864	102224
	М.Ч.О.	408	4912	27728	1524	7704	40880	3276	25864	102224
	В.М.Ч.О.	0.034	0.336	1.790	0.150	1.653	2.140	0.424	5.507	57.41
△EG	М.В.	0.027	0.310	3.503	0.107	1.130	4.650	0.497	4.307	104.534
	Ч.О.М.В.	320	2176	13760	1080	6400	27680	3696	29568	90496
	М.Ч.О.	320	2176	13760	1160	6400	27680	3696	29568	90496
	В.М.Ч.О.	0.027	0.310	3.503	0.087	1.130	4.650	0.497	4.307	104.534
△RR	М.В.	0.037	0.540	3.703	0.180	2.017	16.970	0.890	17.510	226.590
	Ч.О.М.В.	384	2208	14016	1240	6400	24160	3472	30912	103936
	М.Ч.О.	384	2336	14016	1400	6880	27680	4480	30912	103936
	В.М.Ч.О.	0.037	0.410	3.703	0.120	1.287	4.600	0.294	17.510	226.590

М.В. — худшее время выполнения алгоритмов для случайных систем при фиксированных r, m;
Ч.О.М.В. — число операций при выполнении алгоритмов для системы с худшим временем;
М.Ч.О. — максимальное число операций выполнения алгоритмов при фиксированных r, m;
В.М.Ч.О. — время выполнения алгоритмов для системы с максимальным числом операций.

Таблица 2. Сравнение времени работы алгоритмов с опцией remember и без нее

$m=6$		$r=3$	$r=4$	$r=5$	$r=6$	$r=7$	$r=8$	$r=9$
RR	В.	0.690	1.303	2.597	8.950	37.480	50.950	296.633
	Ч.О.Д.	480	1170	2160	3654	5760	8370	11760
RR remember	В.	0.657	1.257	2.593	8.970	37.213	51.550	309.010
	Ч.О.Д.	192	360	576	840	1152	1512	1920
△RR	В.	0.614	1.137	2.330	4.414	8.264	14.810	24.747
	Ч.О.Д.	384	1170	2520	4746	7824	12150	17640
△RR remember	В.	0.596	1.067	2.164	4.144	7.583	13.747	23.933
	Ч.О.Д.	192	630	1224	1974	2880	3942	5160

В. — время выполнения алгоритмов;
Ч.О.Д. — число выполненных операций дифференцирования.

исследование сопряжено с немалыми трудностями (впрочем, известны некоторые исследования битовой сложности, связанные с задачами сходного типа, — см., например, [15]). Понятно, что в процессе применения алгоритмов EG, RR, △EG и △RR происходит "распухание" элементов оператора L. Алгоритмы EG и RR в том виде, как они описаны в пп. 2.3, 2.4, имеют в этом отношении некоторые преимущества: в результате замены строки $L_{i,*}$ строкой (4) или же строкой (6) упомянутое "распухание" элементов может произойти только в одной этой строке. В то же вре-

Таблица 3. Время работы алгоритмов для разреженных систем с $r = 1$, $m = 50$.

$r = 1$ $m = 50$	7%	8%	9%	10%	11%
EG	0.994	36.067	71.540	97.503	177.897
RR	0.010	31.014	22.420	12.717	19.757
△EG	0.020	52.880	43.467	56.773	88.563
△RR	0.010	54.090	45.410	73.053	190.403

мя при приведении выявляющих матриц к треугольному виду (п. 2.5) "распухание" затрагивает значительно большее число элементов, и операции над элементами становятся более дорогостоящими. Поэтому, несмотря на привлекательность оценок (9), (10), алгоритмы EG и RR, как правило, работают быстрее, чем △EG, △RR, что подтверждается экспериментами (см. табл. 1).

Разумеется, надо принимать во внимание, что алгоритмы △EG и △RR выполняют по сравнению с EG и RR дополнительную работу — выявляющая матрица приводится к треугольному виду. Такой результат может оказаться предпочтительнее. В некоторых случаях проявлялось преимущество алгоритмов △EG и △RR и по времени. Например, преимущество △RR перед RR видно по табл. 2. Преимущество △EG перед EG проявилось в экспериментах с разреженными системами невысокого порядка с большим числом уравнений: в табл. 3 приведено время работы алгоритмов для случайно сгенерированных систем порядка $r = 1$ с числом уравнений $m = 50$, число ненулевых коэффициентов составляет 7–11%, коэффициенты — полиномы степени не выше 8.

Несомненное достоинство RR и △RR, как уже было сказано, состоит в том, что в результате получается система, эквивалентная исходной. В ряде случаев RR оказывается более эффективным по времени, чем EG (табл. 3). Но не исключены и случаи, когда более эффективны EG и △EG (табл. 1).

Использование стандартных процедур линейной алгебры в алгоритмах △EG и △RR для приведения выявляющей матрицы к треугольному виду (по аналогии с п. 3.2), изменит порядок роста сложностей алгоритмов △EG, △RR, и вместо m^3 в (9), (10) мы получим $m^{\omega+1}$, что лишит алгоритмы △EG, △RR их обсуждавшегося в п. 2.5 теоретического преимущества перед EG и RR.

5. ЗАКЛЮЧЕНИЕ

Как мы видим, алгоритмы EG и RR допускают изменения, о которых из общих соображений и исходя из оценок сложности можно предположить, что они (эти изменения) приведут к более экономным вариантам этих алгоритмов. Но, как показывают наши эксперименты, проявление этой большей экономности не носит систематического характера. Более того, в целом она проявляется довольно редко. Поэтому основной может быть рекомендация пытаться использовать процедуры `EG` и `RR` (в зависимости от того, какую из выявляющих матриц, ведущую или фронтальную, требуется сделать невырожденной), а уж когда оператор или система за приемлемое время не поддаются соответствующему преобразованию, то пробовать применить другие процедуры пакета `EGRR`; возможно, что это даст желаемый результат.

СПИСОК ЛИТЕРАТУРЫ

1. *Abramov S.A.* On the differential and full algebraic complexities of operator matrices transformations // Proceedings of CASC, 2016. P. 1–14.
2. *van der Put M., Singer M.F.* Galois Theory of Linear Differential Equations // Grundlehren der mathematischen Wissenschaften, 328 Springer, Heidelberg, 2003.
3. *Rosenlicht M.* Integration in finite terms // The American Mathematical Monthly. 1972. V. 79. No. 9. P. 963–972.
4. *Abramov S., Barkatou M.* On the dimension of solution spaces of full rank linear differential systems // Proceedings of CASC 2013. P. 1–9.
5. *Abramov S., Barkatou M.* On solution spaces of products of linear differential or difference operators // ACM Communications in Computer Algebra. 2014. V. 48. I. 4. P. 155–165.
6. *Miyake M.* Remarks on the formulation of the Cauchy problem for general system of ordinary differential equations // Tôhoku Mathematical Journal. 1980. V. 32. No. 1. P. 79–89.
7. *Abramov S., Bronstein M.* On solutions of linear functional systems // Proceedings of ISSAC 2001. P. 1–6.

8. *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

9. *Абрамов С.А., Хмельнов Д.Е.* Особые точки решений линейных обыкновенных дифференциальных систем с полиномиальными коэффициентами // Фундамент. и прикл. матем. 2012. Т. 17(1). С. 3–21.

10. *Beckermann B., Cheng H., Labahn G.* Fraction-free row reduction of matrices of Ore polynomials // J. of Symbolic Computation. 2006. V. 41. I. 5. P. 513–543.

11. *Barkatou M., El Bacha C., Labahn G., Pflügel E.* On simultaneous row and column reduction of higher-order linear differential systems // J. of Symbolic Computation. 2013. V. 49. P. 45–64.

12. *Abramov S.* EG-eliminations // J. of Difference Equations and Applications. 1999. V. 5. I. 4–5. P. 393–433.

13. *Mulders T., Storjohann A.* On Lattice Reduction for Polynomial Matrices // J. of Symbolic Computation. 2003. V. 35. I. 4. P. 377–401.

14. Maple online help // http://www.maplesoft.com/support/help/

15. *Giesbrecht M., Sub Kim M.* Computing the Hermite form of a matrix of Ore polynomials // J. of Algebra. 2013. V. 376. P. 341–362.