Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Тощев Андрей Александрович

Разработка и реализация эвристических методов улучшения характеристик одного алгоритма компьютерной алгебры

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

Панфёров Антон Александрович

Москва, 2017

Аннотация

В работе исследуется один из алгоритмов компьютерной алгебры — алгоритм Row-Reduction. Он предназначен для приведения операторной матрицы по строкам. Алгоритм анализируется на всевозможные неоднозначности. Приводятся примеры работы алгоритма с его возможными неоднозначностями и уделяется внимание разработке эвристических методов улучшения характеристик результатов работы алгоритма. В результате исследований была разработана система правил для разрешения неоднозначного выбора, возникающего в алгоритме Row-Reduction. На основе сформулированных правил в системе компьютерной алгебры Maple была реализована специальная версия алгоритма Row-Reduction. Эксперименты, проведённые с программой, показали, что предложенные правила в 70.5% случаев позволяют получать приведённые матрицы с более хорошими характеристиками. Использование модифицированной версии алгоритма позволит сократить вычислительные затраты в алгоритмах, использующих Row-Reduction как составную часть.

Содержание

Введение

Компьютерная алгебра — одна из областей компьютерных наук, где разрабатываются эффективные алгоритмы для решения алгебраических задач. В компьютерной алгебре предполагается, что и исходные данные, и результаты решения представлены в символьном виде [1, с. 7]. *Системой компьютерной алгебры* называют специальное программное обеспечение для символьных вычислений, то есть для выполнения преобразований и работы с математическими выражениями в аналитической форме. Система содержит процедуры выполнения базовых преобразований выражений и набор пакетов процедур, предназначенных для решения более специальных задач.

В научных исследованиях и технических расчетах специалистам приходится гораздо больше заниматься преобразованиями формул, чем численным счетом. С появлением ЭВМ основное внимание уделяется автоматизации численных вычислений, ЭВМ ещё в 50-х годах прошлого века начали применяться для решения таких задач, как символьные преобразования, например, символьное дифференцирование.

Первые системы компьютерной алгебры (СКА / CAS) начали разрабатываться в конце 60-х годов. С тех пор создано множество различных систем, получивших различную степень распространения: Maple, MATHLAB, Mathematica, MUPAD и другие. Все они имеют широкое применение в научно-технических областях, где под каждую область может быть создано несколько систем компьютерной алгебры, которые различаются разными способами решения одних и тех же задач. В системах компьютерной алгебры общего назначения, как правило, используется собственный язык программирования и библиотеки алгоритмов для написания собственных программ. Главной причиной дальнейших разработок СКА остаётся [2, с. 4, 20-22]:

1. простота их использования;
2. скорость работы;
3. широкий функционал;
4. язык программирования и интерпретатор;
5. управление памятью, включая сборщик мусора;
6. большая библиотека математических алгоритмов.

Рассмотрим некоторые исторически значимые системы компьютерной алгебры, которые используются в настоящее время [3, с. 7], [4].

Macsyma является системой компьютерной алгебры, которая разрабатывалась с 1968 гг. по 1982 гг. в MIT как научный проект MAC (Mathematics and Computation), а затем стала продаваться как коммерческий продукт Maxima. Это первая символическая система математики общего назначения и одна из самых ранних систем, основанная на математических знаниях, часть которых легла в основу Mathematica, Maple и других систем.

Reduce — система компьютерной алгебры общего назначения, созданная физиком А. Херном (Anthony C. Hearn) в 1960-х годах для применения в физике; первая распространяемая версия появилась в 1968 г., а последняя версия создана в 2009 г. С 2008 г. система распространяется бесплатно.

Maple разрабатывается с 1980 года сотрудниками Symbolic Computation Group в Университете Waterloo (первая версия появилась в 1984 году). С 1988 года система продается компанией Maplesoft. Последняя версия «Maple 2016» была выпущена 22 апреля 2016 года.

Maple состоит из небольшого ядра, написанного на языке программирования Си, которое обеспечивает функционирование языка Maple. Функциональность системы основана на различных библиотеках. Алгоритмы Maple используют численные данные в разных форматах. Символические выражения хранятся в памяти как направленные ациклические графы. Стандартный пользовательский и вычислительный интерфейс реализованы на Java.

Mathematica — система компьютерной алгебры общего назначения, используется во многих научных, инженерных, математических и вычислительных областях. Система была задумана Стивеном Вольфрамом (физик, математик и программист) и разработана компанией Wolfram Research (Шампейн, штат Иллинойс, США). Начало разработки — 1986 г.; первая версия — 1988 г.; последняя версия: «Mathematica 11.1» – март 2017 г..

Wolfram Alpha — база знаний и набор вычислительных алгоритмов, выстроенных как вопросно-ответная система. Это онлайн сервис, доступный на сайте `www.wolframalpha.com`, который был запущен 15 мая 2009 года Стивеном Вольфрамом. Система не является поисковой, как, например, Google, однако она способна обрабатывать поисковые запросы и символьные вычисления, которые также может обрабатывать любая из вышеперечисленных систем компьютерной алгебры.

Math Partner — это ещё один сервис, представляющий собой систему компьютерной алгебры с web-интерфейсом, где масштабные математические объекты вычисляются

на многопроцессорном вычислительном кластере. Кроме того, MathPartner предоставляет возможность загружать и исполнять на кластере пользовательские программы, которые были предварительно загружены с помощью web-интерфейса `mathpar.cloud.unihub.ru` [5].

Таким образом, системы компьютерной алгебры позволяют работать над математическими выражениями так же как, если бы с ними пришлось работать вручную. Они предназначены для различных задач из разных областей, и могут быть частью онлайн сервисов.

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. Один из таких вспомогательных алгоритмов рассмотрен дальше.

Пусть $K = \mathbb{Q}$ — поле рациональных чисел, а $K(x)$ — поле рациональных функций с коэффициентами из K с производной $\partial = '$:

$$\forall a \in K(x), \partial a = \frac{da}{dx} + a\partial. \tag{1}$$

Рассмотрим систему дифференциальных уравнений над $K(x)$. Кольцо $n \times n$ матриц с элементами в некотором кольце (поле) R обозначается через $\mathrm{Mat}_n(R)$. Если L — матрица, то обозначим через $L_{i,*}$ её i-ю строку , $L_{*,j}$ —j-й столбец .

Рассмотрим систему вида:

$$A_r y^{(r)} + A_{r-1} y^{(r-1)} + ... + A_1 y' + A_0 y = 0, \tag{2}$$

где A_i — квадратные матрицы размера $n \times n$, $A_i \in Mat_n(K(x))$ $(A_r \neq 0)$, y — вектор неизвестных функций от x: $y = (y_1, ..., y_n)^T$.

Кроме того, если $n = 1$ (матрица, состоящая из одного элемента), то оператор L матриц A_i становится скалярным оператором, имеющий вид полинома от ∂ [6, с. 8]. Такие операторы работают с арифметическими операциями как обычные полиномы, кроме операции умножения. Она в кольце дифференциальных операторов $K(x)[\partial]$ над полем рациональных функций с коэффициентами из $K(x)$ определяется по правилу (1).
Пример 1. $L = a_0 + a_1\partial + \ldots + a_n\partial^n \in K(x)[\partial] \Rightarrow L(x+1) = (a_0 + a_1\partial + \ldots + a_n\partial^n)(x+1) = a_0(x+1) + a_1$.

Систему дифференциальных уравнений (2) произвольного порядка можно представить в виде:

$$L(y) = 0, \tag{3}$$

где L — это матрица дифференциальных операторов (операторная матрица), т. е. матрица, элементами которой являются дифференциальные операторы из кольца $\mathbb{Q}(x)[\partial]$:

$$L = A_r\partial^r + A_{r-1}\partial^{r-1} + \ldots + A_0 \in Mat_n(K(x)[\partial]). \tag{4}$$

Систему (2) можно записать как систему из n скалярных линейных уравнений:

$$L_{i,*}(y) = 0$$

или

$$L_1(y_1, \ldots, y_n) = 0, \ \ldots, \ L_n(y_1, \ldots, y_n) = 0, \tag{5}$$

где

$$L_i(y_1, \ldots, y_n) = \sum_{j=1}^{n} l_{ij}(y_j), l_{ij} \in K(x)[\partial], \quad i, j = 1, \ldots, n, \quad \max_{i,j} ord(l_{ij}) = r.$$

Система (3) имеет полный ранг [7, sec. 2], если уравнения (5) независимы над $K(x)[\partial]$, или, другими словами, строки

$$l_i = (l_{i1}, \ldots, l_{in}), \tag{6}$$

$i = 1, \ldots, n$, линейно независимы над $K(x)[\partial]$. Ранг s системы (3) или матрицы (4) равен числу линейно независимых строк (6).

Стандартный метод решения систем вида (3) основан на сведении исходной системы к системе более простой формы, где, во время преобразований, системы полного ранга сводятся к системам с невырожденной фронтальной матрицей (см. определение 5 из разд. 1.1.3). В компьютерной алгебре существует немало алгоритмов для преобразования и решения линейных дифференциальных систем. В том числе есть такие алгоритмы, которые настраиваются под конкретный случай с операторами разных видов [8]: дифференциальные, разностные и пр. Все эти случаи могут быть описаны с использованием полиномов Оре [9]. Это даёт возможность при реализации алгоритмов абстрагироваться от конкретного случая, делая алгоритм настраиваемым. Одним из таких алгоритмов является алгоритм Row-Reduction (RR) [10].

1 Алгоритм Row-Reduction

Прежде чем перейти к постановке задачи данной работы, рассмотрим алгоритм Row-Reduction. Алгоритм Row-Reduction — это алгоритм приведения по строкам операторной матрицы. Предложенный в статье «Fraction-free row reduction of matrices of ore polynomials» Б. Беккерманом, Г. Лабаном и Х. Ченгом [10], алгоритм RR был сформулирован в обобщённом виде для работы с матрицами, элементами которых являются полиномы Оре. Однако в настоящей работе внимание сосредоточено на дифференциальном случае.

1.1 Дифференциальные операторные матрицы над полем рациональных функций

1.1.1 Линейные дифференциальные операторы и системы

Определение 1. Ненулевая матрица дифференциальных операторов (операторная матрица) L размера $m \times n$ может быть записана в виде (4), где $r \in \mathbb{N}$, $A_i \in K(x)^{m\times n}$ для $i = 0, \ldots, r$ и $A_r \neq 0$. Число r называется порядком L и обозначается $ord(L)$. Матрица $A_r(x)$ называется *ведущей* матрицей L и обозначается $lc(L)$. Если $L = 0$, мы полагаем $ord(L) = -\infty$ и $lc(L) = 0$.

Определение 2. Пусть $L \in K(x)[\partial]^{m\times n}$ и $J \subseteq \{1, \ldots, m\}$. Строки $L_{i,*}$ с индексами $i \in J$ называются $K(x)[\partial]$-линейно зависимыми, если существуют дифференциальные операторы $\{W_i\}_{i\in J} \subset K(x)[\partial]$, не все одновременно равные нулю, и такие, что $\sum_{i\in J} W_i L_{i,*} = 0$; иначе, они $K(x)[\partial]$-линейно независимы.

1.1.2 Унимодулярные операторы и эквивалентность операторных матриц

Определение 3. Операторная матрица $U \in \mathrm{Mat}_m(K(x)[\partial])$ называется унимодулярной (или обратимой), если существует такая операторная матрица $\overline{U} \in \mathrm{Mat}_m(K(x)[\partial])$, что $\overline{U}U = U\overline{U} = I_m$, где I_m — единичная матрица размера $m \times m$.

Примерами унимодулярных матриц, обратных по отношению друг к другу, являются следующие матрицы: $\left(\begin{smallmatrix} 1 & \partial \\ 0 & 1 \end{smallmatrix}\right)$, $\left(\begin{smallmatrix} 1 & -\partial \\ 0 & 1 \end{smallmatrix}\right)$.

Существует три типа элементарных преобразований строк матрицы дифференциальных операторов L:

1. перестановка двух строк;

2. умножение строки слева на ненулевой элемент из $K(x)$;

3. сложение одной строки операторной матрицы с другой строкой, умноженной слева на некоторый дифференциальный оператор с коэффициентами из $K(x)$.

Утверждение 1. [10, Lemma A.3] Пусть L матрица дифференциальных операторов размера $m \times n$ и U и V две унимодулярные матрицы дифференциальных операторов размеров $m \times m$ и $n \times n$ соответственно. Тогда ранги L, UL и LV одинаковы.

Определение 4. Две операторные матрицы $\breve{L}, L \in K(x)[\partial]^{m\times n}$ эквивалентны, если существуют унимодулярные матрицы $U \in K(x)[\partial]^{m\times m}$ и $V \in K(x)[\partial]^{n\times n}$ такие, что $\breve{L} = ULV$.

1.1.3 Фронтальная матрица

Пусть операторная матрица полного ранга $L \in \mathrm{Mat}_m(K[\partial])$ имеет вид (4). Если $i = 1, \ldots, m$, то определим $\alpha_i(L)$ как наибольшее целое k, $0 \le k \le r$, при котором i-я строка матрицы представления (4), т. е. строка $(A_k)_{i,*}$, ненулевая, т. е. $\alpha_i(L)$ — максимальный порядок дифференцирования i-й строки $L_{i,*}$.

Определение 5. Число α_i будем называть *порядком i-й строки* операторной матрицы L и *порядком i-го уравнения* системы $L(y) = 0$.

Определение 6. Матрица $M \in \mathrm{Mat}_m(K)$, такая, что $M_i = (A_{\alpha_i(L)})_i, i = 1, \ldots, m$, называется *фронтальной матрицей* операторной матрицы L и системы $L(y) = 0$.

Определение 7. Матрица дифференциальных операторов называется *приведённой по строкам*, если ненулевые строки её фронтальной матрицы линейно независимы.

Рассмотрим пример, поясняющий определения и утверждения, представленные выше.

Пример 2. Оператор

$$L = \begin{bmatrix} \partial^3 + x & 2\partial^2 & x^2 + x \\ \partial^2 & x\partial^2 & 2x^2 + 1 \\ \partial & x\partial & 1 \end{bmatrix} \tag{7}$$

имеет развернутое матричное представление

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \partial^3 + \begin{bmatrix} 0 & 2 & 0 \\ 1 & x & 0 \\ 0 & 0 & 0 \end{bmatrix} \partial^2 + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & x & 0 \end{bmatrix} \partial + \begin{bmatrix} x & 0 & x^2 + x \\ 0 & 0 & 2x^2 + 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Система $L(y) = 0$ с оператором L может быть представлена в виде системы уравнений:

$$\begin{cases} y_1''' + 2y_2'' + xy_1 + (x^2 + x)y_3 = 0 \\ y_1'' + xy_2'' + (2x^2 + 1)y_3 = 0 \\ y_1' + xy_2' + y_3 = 0 \end{cases}$$

где y_1, y_2 и y_3 — это компоненты вектора y: $y = (y_1, y_2, y_3)^T$

Ведущей матрицей L и системы $L(y) = 0$ служит $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.

Фронтальной матрицей (7) и соответствующей ему системы служит

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & x & 0 \\ 1 & x & 0 \end{bmatrix}.$$

1.2 Алгоритм Row-Reduction

Алгоритм RR преобразует матрицу дифференциальных операторов к приведённому по строкам виду.

Нужно отметить, что до этого мы работали с операторами над полем рациональных функций $K(x)$. Теперь от поля перейдём к кольцу полиномов $K[x]$. Перейти от рациональных коэффициентов к полиномам можно, помножив каждую строку операторной матрицы на наименьшее общее кратное знаменателей. В дальнейшем мы будем работать в кольце полиномов $K[x]$.

Теорема 1. [10, Th. 2.2] Пусть $L \in K[x][\partial]^{m \times n}$ — матрица дифференциальных операторов ранга $s \leq \min(m, n)$. Тогда существует такая унимодулярная матрица $U \in K(x)[\partial]^{m \times m}$, что

$$UL = \left[\begin{array}{c} L^* \\ \hline 0 \end{array}\right], \tag{8}$$

где L^* — матрица дифференциальных операторов размера $s \times n$, привидённая по строкам, у которой $ord(L^*) \leq ord(L)$ и все строки ненулевые.

Утверждение 2. [10] Ранг приведённой по строкам операторной матрицы равен рангу фронтальной матрицы.

Алгоритм Row-Reduction[11, с. 49].

Вход: $L_0 \in K[x][\partial]^{m\times n}$ — операторная матрица с порядками строк $\delta = (\delta_1, \ldots, \delta_m)$.

Выход: Операторная матрица $L \in K[x][\partial]^{m\times n}$, приведенная по строкам, и унимодулярный оператор $U \in K[x][\partial]^{m\times m}$ такой, что $L = UL_0$.

Инициализация: Пусть $L = L_0$, F_0 — фронтальная матрица L, $\delta = (\delta_1, \ldots, \delta_m)$ — вектор порядков строк L и $U = I_m$.

WHILE ненулевые строки F_0 линейно зависимы **DO**

1. Вычисляем $\nu = (\nu_1, \ldots, \nu_m) \in K[x][\partial]^{1\times m} \setminus 0$ вектор коэффициентов линейной зависимости ненулевых строк F_0.
2. Выбираем число k такое, что $\nu_k \neq 0$ и $\delta_k = \max(\delta_i;\ \nu_i \neq 0)$.
3. Заменяем $L_{k,*} \leftarrow \sum_{i=1}^{m} \nu_i \partial^{\delta_k - \delta_i} L_{i,*}$.
4. Заменяем $U_{k,*} \leftarrow \sum_{i=1}^{m} \nu_i \partial^{\delta_k - \delta_i} U_{i,*}$.
5. Обновляем L и δ.

END DO;

Return L и U;

В описании работы алгоритма RR можно заметить, что для заданной операторной матрицы её приведённый по строкам вид различается, потому что в приведённом алгоритме есть «развилки», на которых приходится выбирать, по какому пути продолжать работу.

Пример 3. Рассмотрим пример исходной операторной матрицы:

$$P = \begin{bmatrix} (4x+9)\partial + 1 & 0 & (8x-11)\partial - 5 \\ (-40x^2 - 66x + 54)\partial + 63x^2 - 88x + 30 & 7x - 4 & (-80x^2 + 144x - 58)\partial + 50x - 30 \\ 0 & 0 & 1 \end{bmatrix}$$

Фронтальная матрица для P имеет вид:

$$\begin{bmatrix} 4x+9 & 0 & 8x-11 \\ -40x^2 - 66x + 54 & 0 & -80x^2 + 144x - 58 \\ 0 & 0 & 1 \end{bmatrix}$$

Фронтальная матрица вырождена и имеет один линейно независимый вектор ν коэффициентов линейной зависимости строк фронтальной матрицы:

$$\left\{ \begin{bmatrix} 10x - 6 & 1 & 14x - 8 \end{bmatrix}^T \right\}.$$

Поскольку у операторной матрицы P вектор порядков строк $\delta = (1, 1, 0)$ имеет два максимальных одинаковых порядка дифференцирования строк, равных 1, при $\nu_1 = 10x - 6 \neq 0$ и $\nu_2 = 1 \neq 0$, то в результате работы алгоритма можно получить две приведённые матрицы.

Результирующая матрица при выборе первой строки:

$$\begin{bmatrix} 9x - 6 & 1 & 0 \\ (-40x^2 - 66x + 54)\partial + 63x^2 - 88x + 30 & 7x - 4 & (-80x^2 + 144x - 58)\partial + 50x - 30 \\ 0 & 0 & 1 \end{bmatrix}$$

Результирующая матрица при выборе второй строки:

$$\begin{bmatrix} (4x+9)\partial + 1 & 0 & (8x-11)\partial - 5 \\ 9x - 6 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2 Постановка задачи

Целью выпускной работы является анализ и разработка методов по разрешению неоднозначностей, возникающих в ходе работы алгоритма RR, с целью улучшения характеристик результирующей матрицы.

Для достижения поставленной цели необходимо решить следующие задачи:

- Проанализировать алгоритм RR и выявить все возможные неоднозначности, возникающие в его работе, а также проанализировать природу этих неоднозначностей и возможность целенаправленного их разрешения для улучшения характеристик получаемых результатов.
- Проанализировать вид всех возможных результирующих матриц с целью выработки критериев сравнения и сформулировать эвристические правила разрешения неоднозначного выбора в ходе алгоритма RR, позволяющие получать результирующие операторные матрицы с хорошими характеристиками.
- Реализовать специальную версию RR, использующую предложенные правила.
- Провести с разработанной версией RR ряд экспериментов для оценки предложенных эвристик.

Разработанная улучшенная версия RR может быть в дальнейшем использована в других известных алгоритмах компьютерной алгебры, использующих приведение по строкам. Например в таких, как алгоритм Simultaneously Row and Column reduction[11], алгоритм Extract[12], алгоритм $\Delta - RR$[6].

3 Недетерминированность алгоритма Row-Reduction

3.1 Выявление неоднозначностей алгоритма Row-Reduction

Для того, чтобы выявить неоднозначности алгоритма RR, приведём примеры операторных матриц, которые проиллюстрируют возникновение неоднозначностей во время его работы. Мы выбрали две операторные матрицы (см. (9) и (12)). Для начала исследуем матрицу (9).

3.1.1 Выбор вектора коэффициентов линейной зависимости строк вырожденной фронтальной матрицы

На шаге 1 алгоритма RR строится вектор коэффициентов линейной зависимости строк фронтальной матрицы. Но этот вектор определяется неоднозначно.

Пример 4. Рассмотрим операторную матрицу:

$$\begin{bmatrix} \partial^4 + x\partial & 2\partial^3 & (x^2+x)\partial \\ \partial^3 + x & 2\partial^2 & x^2 - x \\ \partial^2 & x\partial & 1 \end{bmatrix} \tag{9}$$

Её фронтальная матрица имеет вид:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{10}$$

Очевидно, что фронтальная матрица является вырожденной. Найдём линейно независимые векторы линейной зависимости её строк:

$$\left\{ \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T, \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T \right\} \tag{11}$$

Мы видим, что количество линейно независимых векторов коэффициентов линейной зависимости строк фронтальной матрицы может быть больше одного. Значит, необходимо выбирать с каким из векторов будет производиться дальнейшая работа алгоритма RR.

3.1.2 Выбор элемента вектора коэффициентов линейной зависимости строк фронтальной матрицы дифференциального оператора

На шаге 2 алгоритма RR среди строк матрицы дифференциальных операторов, соответствующих ненулевым элементам вектора полученного на шаге 1, необходимо вы-

брать строку с наибольшим порядком дифференцирования. В случае наличия нескольких таких строк, появляется очередная неоднозначность выбора.

Пример 5. Рассмотрим матрицу дифференциальных операторов и найдем векторы линейной зависимости строк фронтальной матрицы:

$$\begin{bmatrix} 7\partial^3+3\partial^2+\partial & 11\partial^2 & 2x^2+1 & 1 \\ x\partial^2+x & (x^2+x)\partial & 6\partial^2 & 1 \\ 3\partial & 3\partial^3+x & 1 & 1 \\ 3\partial^3 & x & 1 & 1 \end{bmatrix} \tag{12}$$

1, 3 и 4-я строки матрицы дифференциальных операторов (12) имеют одинаковый порядок дифференцирования, равный трём, а 2-я строка — порядок два.

Вектор коэффициентов линейной зависимости строк фронтальной матрицы для (12):

$$\begin{bmatrix} -3/7 & 0 & 0 & 1 \end{bmatrix}^T \tag{13}$$

Ненулевыми компонентами являются $-3/7$ и 1, соответствующие первой и четвёртой строкам. Обе строки имеют порядок 4, а значит для дальнейшей работы алгоритма может быть выбрана любая из них.

Таким образом, рассмотрим список унимодулярных матриц для строк с одинаковым наибольшим порядком дифференцирования, полученных на шаге 4 алгоритма RR, где исходной операторной матрицей является матрица (12):

$$\begin{bmatrix} -3 & 0 & 0 & 7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 0 & 0 & 7 \end{bmatrix} \tag{14}$$

Из списка матриц (14) мы видим, что унимодулярные операторы различаются. Следовательно, возможны различные варианты результирующей матрицы после выполнения алгоритма RR.

Таким образом, мы выявили две неоднозначности алгоритма, для решения которых мы разработаем правила разрешения неоднозначностей алгоритма RR.

3.2 Средства разработки модификаций алгоритма RR

Определим инструментальные средства программирования для дальнейшего написания комплекса программ по изучению и разрешению неоднозначностей алгоритма Row-Reduction.

В качестве инструментальной среды будет использована система компьютерной алгебры Maple. Изначально алгоритм Row-Reduction формулируется для матриц, элементами которых являются полиномы Ope. В системе компьютерной алгебры Maple имеются пакеты для работы с дифференциальными операторами. Например, пакеты DEtools, OreTools содержат процедуры для работы с дифференциальными операторами и их системами. OreTools проще в использовании, поэтому для представления дифференциальных операторов мы выбрали средства, предоставляемые им. Пакет OreTools предоставляет структуру *OrePoly*, представляющую скалярные операторы; структура соответствует внутреннему представлению некоммутативных полиномов Ope.

Покажем как выглядят скалярные дифференциальные операторы в виде структуры *OrePoly* в системе Maple. Для этого приведём пример полинома Ope и операторной матрицы в алгебраическом виде и в виде, который использует структуру *OrePoly*.

Пример 6. Дифференциальный оператор $a_r\partial^r + a_{r-1}\partial^{r-1} + ... + a_1\partial + a_0$ с использованием *OrePoly* представляется в виде: $OrePoly(a_0, a_1, \ldots, a_r)$.

Операторная матрица (7) из примера 2 представляется в Maple в виде:

$$L = \begin{bmatrix} OrePoly(x,0,0,1) & OrePoly(0,0,2) & OrePoly(x^2+x) \\ OrePoly(0,0,1) & OrePoly(0,0,x) & OrePoly(2x^2+1) \\ OrePoly(0,1) & OrePoly(0,x) & OrePoly(1) \end{bmatrix}$$

Дальнейшие реализации программ написаны на языке Maple и в качестве подключаемых библиотек задействованы пакеты из системы компьютерной алгебры Maple.

4 Эвристические методы разрешения неоднозначностей

4.1 Сравнительный анализ различных результирующих матриц

Рассмотрим все итоговые результирующие матрицы при выполнении алгоритма RR на примере разных операторных матриц, у которых ненулевые строки фронтальной матрицы линейно зависимы. Для получения результирующих матриц была написана процедура *outputRR* на языке Maple со специальной реализацией алгоритма RR, которая собирает в единую структуру все результирующие матрицы, полученные в результате отработки алгоритмом RR всех возможных сценариев преобразования исходной матрицы.

Для пошаговой наглядности, вывод данной программы представляет собой список списков, каждый элемент которого — это последовательность результирующих матриц, полученных на каждой новой итерации RR. Первый элемент этого списка — исходная матрица дифференциальных операторов, а последний — матрица в приведённом по строкам виде. Подробно рассмотрим работу алгоритма RR и все её выводы, за исключением шага 4, на котором производится замена строки в унимодулярной матрице, которая на данном этапе нас не интересует.

Пример 7. Рассмотрим работу алгоритма, во время которой возникает *неоднозначный выбор элемента вектора коэффициентов линейной зависимости строк фронтальной матрицы* операторной матрицы, на примере операторной матрицы размерности 3×3:

$$\begin{bmatrix} \partial^3 + x & 2\partial^2 & x^2 + x \\ \partial & x\partial^2 & 2x^2 + 1 \\ \partial^2 & x\partial & 1 \end{bmatrix} \tag{15}$$

Из вывода программы (см. рисунок 1) мы видим, что неоднозначность проявляется на 2-й итерации, т. к. все строки операторной матрицы имеют максимальный порядок дифференцирования, равный трём, а вектор коэффициентов линейной зависимости строк фронтальной матрицы равен $\left\{ \begin{bmatrix} -x & x-2 & 0 \end{bmatrix}^T \right\}$. Поэтому, на момент неоднозначности алгоритма произведём замену 1-й и 2-й строк результирующей матрицы, полученной на 2-й итерации. Для этого, на 2-й итерации алгоритма у результирующей матрицы произведём замену 1-й и 2-й строк (шаг 3 алгоритма RR), чтобы получить матрицы в приведённом по строкам виде, которые впоследствии назовём итоговыми

результирующими матрицами.

Первый элемент — список списков:

$$\begin{bmatrix} \partial^3+x & 2\partial^2 & x^2+x \\ \partial & x\partial^2 & 2x^2+1 \\ \partial^2 & x\partial & 1 \end{bmatrix}, \begin{bmatrix} -x & (x-2)\partial^2+\partial & \partial-x^2-x \\ \partial & x\partial^2 & 2x^2+1 \\ \partial^2 & x\partial & 1 \end{bmatrix}, \begin{bmatrix} (x-2)\partial+x^2 & -x\partial & -x\partial+3x^3-3x^2+x-2 \\ \boxed{\begin{matrix}\partial & x\partial^2 & 2x^2+1\end{matrix}} & & \\ \partial^2 & x\partial & 1 \end{bmatrix}$$

Второй элемент — список списков:

$$\begin{bmatrix} \partial^3+x & 2\partial^2 & x^2+x \\ \partial & x\partial^2 & 2x^2+1 \\ \partial^2 & x\partial & 1 \end{bmatrix}, \begin{bmatrix} -x & (x-2)\partial^2+\partial & \partial-x^2-x \\ \partial & x\partial^2 & 2x^2+1 \\ \partial^2 & x\partial & 1 \end{bmatrix}, \begin{bmatrix} \boxed{\begin{matrix}-x & (x-2)\partial^2+\partial & \partial-x^2-x\end{matrix}} & & \\ (x-2)\partial+x^2 & -x\partial & -x\partial+3x^3-3x^2+x-2 \\ \partial^2 & x\partial & 1 \end{bmatrix}$$

Рис. 1. Список результатов *List1* работы RR для (15)

Обозначим результат программы $List1$, тогда $List1[i]$ — это i-й путь возможной работы алгоритма RR. Выбирая i-й путь, матрица $List1[i][j]$ была получена на $(j-1)$-й итерации. 3-й параметр k в $List1[i][j][k]$ представляет k-ую строку матрицы $List1[i][j]$.

Подробно рассмотрим работу алгоритма RR по всем итерациям.

1-я итерация алгоритма.

Шаг 1. Фронтальная матрица операторной матрицы (15) имеет вид:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & x & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{16}$$

Вектор коэффициентов линейной зависимости строк фронтальной матрицы (16) имеет вид $\nu = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$. Других векторов, которые были бы линейно независимы с вектором ν, нет.

Шаг 2. Вектор порядков строк $\delta = (3, 2, 2)$. Если ненулевые значения элементов вектора ν соответствуют 1-й и 3-й строкам, то выберем первую строку, поскольку её порядок дифференцирования больше порядка дифференцирования 3-й строки.

Шаг 3. Заменим 1-ю строку операторной матрицы по формуле:

$$L_{1,*} \leftarrow \sum_{i=1}^{3} \nu_i \partial^{3-\delta_i} L_{i,*} \tag{17}$$

Окончательная формула для 1-й строки следующая:

$$L_{1,*} \leftarrow -L_{1,*} + \partial L_{3,*}$$

2-я итерация алгоритма. Получена результирующая матрица $List1[1][2]$ (= $List1[2][2]$) с новым вектором порядков строк $\delta = (2, 2, 2)$ и новой фронтальной матрицей:

$$\begin{bmatrix} 0 & x-2 & 0 \\ 0 & x & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Шаг 1. Вектор линейной зависимости строк для новой фронтальной матрицы:

$$\left\{ \begin{bmatrix} -x & x-2 & 0 \end{bmatrix}^T \right\} \tag{18}$$

Шаг 2. Ненулевые значения элементов вектора (18) стоят на 1-й и 2-й строках. Произведём замену, предусмотренную шагом 3 алгоритма, по 1-й или 2-й строкам, т. к. порядки δ_1 и δ_2 равны 2. Значит, выбор строк операторной матрицы неоднозначен, и можно получить две новые результирующие матрицы.

Шаг 3. После замены строк по формулам:

$$L_{1,*} \leftarrow -xL_{1,*} + (x-2)L_{2,*},$$

$$L_{2,*} \leftarrow -xL_{1,*} + (x-2)L_{2,*},$$

были получены новые результирующие матрицы $List1[1][3]$ и $List1[2][3]$ соответственно.

Заметим, что у новых результирующих матриц фронтальные матрицы невырождены. Значит, работу алгоритм закончил.

Сравним итоговые результирующие матрицы в списке (см. рисунок 1), найдем их сходства и различия. Сходства строк $List1[1][3][2]$ и $List1[2][3][1]$:

- максимальный порядок дифференциирования строк равен 2;
- максимальная степень полиномов в коэффициентах ненулевых порядков дифференцирования строк равна 2;
- также заметим, что одинаковые строки могут различаться с точностью до коэффициента.

Различия строк $List1[1][3][2]$ и $List1[2][3][1]$:

- суммарное количество слагаемых в коэффициентах дифференциальных операторов элементов строки, равно 4 и 7 соответственно;
- сумма порядков дифференцирования элементов строки, равна 3 и 4 соответственно;
- количество степеней диффренцирования в строке, равно 2 и 3 соответственно;

- сумма степеней полиномов в коэффициентах при порядках дифференцирования элементов строки, равна 3 и 5 соответственно.

Также, если сравнивать элементы различных строк, имеющих одинаковый порядок столбцов результирующих матриц, то результаты сравнения элементов могут различаться.

Пример 8. Рассмотрим вывод программы, во время работы которой возникает неоднозначный выбор вектора коэффициентов линейной зависимости строк фронтальной матрицы, на примере ранее рассмотренной операторной матрицы (9).

На выводе программы (см. рисунок 2) многозначность появляется только на 1-й итерации в виде двух векторов (11) линейной зависимости строк фронтальной матрицы.

Первый элемент — список списков:

$$\begin{bmatrix} \partial^4+x\partial & 2\partial^3 & (x^2+x)\partial \\ \partial^3+x & 2\partial^2 & x^2-x \\ \partial^2 & x\partial & 1 \end{bmatrix}, \quad \begin{bmatrix} -x\partial & (x-2)\partial^3+2\partial^2 & \partial^2-(x^2+x)\partial \\ \partial^3+x & 2\partial^2 & x^2-x \\ \partial^2 & x\partial & 1 \end{bmatrix},$$

$$\begin{bmatrix} -x\partial & (x-2)\partial^3+2\partial^2 & \partial^2-(x^2+x)\partial \\ -x & (x-2)\partial^2+\partial & \partial-x^2+x \\ \partial^2 & x\partial & 1 \end{bmatrix}, \quad \begin{bmatrix} -1 & 0 & 2x\partial-2x+1 \\ -x & (x-2)\partial^2+\partial & \partial-x^2+x \\ \partial^2 & x\partial & 1 \end{bmatrix}$$

Второй элемент — список списков:

$$\begin{bmatrix} \partial^4+x\partial & 2\partial^3 & (x^2+x)\partial \\ \partial^3+x & 2\partial^2 & x^2-x \\ \partial^2 & x\partial & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & -2x\partial+2x-1 \\ \partial^3+x & 2\partial^2 & x^2-x \\ \partial^2 & x\partial & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & -2x\partial+2x-1 \\ -x & (x-2)\partial^2+\partial & \partial-x^2+x \\ \partial^2 & x\partial & 1 \end{bmatrix}$$

Рис. 2. Список результатов *List2* работы RR для (9)

Теперь рассмотрим работу алгоритма RR на 1-й итерации, т. к. на последующих итерациях неоднозначности не возникают.

1-я итерация алгоритма.

Шаг 1. Для операторной матрицы (9) вектор порядка дифференцирования строк имеет вид $\delta = (4, 3, 2)$, а фронтальная матрица имеет вид (10), для которой два вектора коэффициентов линейной зависимости строк имеют вид:

$$\left\{ \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T, \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T \right\}. \tag{19}$$

Шаг 2. Неоднозначности выбора строк нет, т. к. все порядки строк δ_i различные. Значит, на 1-й итерации есть два возможных пути работы алгоритма RR.

По вектору $\nu = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ максимальный порядок строки

$$\max(\delta_i), \text{ где } \nu_i \neq 0 \text{ при } 1 \leq i \leq 3,$$

равен 4.

Шаг 3. Тогда произведём замену 1-й строки по формуле

$$L_{1,*} \leftarrow -L_{1,*} + \partial^2 L_{3,*},$$

и получим результирующую матрицу, соответствующую элементу $List2[1][2]$.

Шаг 1. Аналогично, найдём результирующую матрицу по вектору $\nu = \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}^T$.

Шаг 2. Максимальный порядок дифференцирования строки $\max(\delta_i) = 4$, где $\delta = (4, 3, 2)$ при $\nu_i \neq 0$.

Шаг 3. Значит, замена произойдёт в 1-й строке по формуле

$$L_{1,*} \leftarrow -L_{1,*} + \partial L_{2,*},$$

и мы получим результирующую матрицу, соответствующую элементу $List2[2][2]$.

Заметим, что помимо сравнения итоговых результирующих матриц необходимо сравнивать результирующие матрицы, т. е. проводить локальный анализ неоднозначного выбора работы алгоритма RR на текущей итерации.

Для того, чтобы сверить полученные результирующие матрицы между собой, достаточно сравнить их изменённые строки $List2[1][2][1]$ и $List2[2][2][1]$, т. к. остальные строки одинаковы.

Таблица 1 содержит критерии сравнения строк:

- $List2[1][2][1] = \begin{bmatrix} -x\partial & (x-2)\partial^3 + 2\partial^2 & \partial^2 - (x^2+x)\partial \end{bmatrix}$;
- $List2[2][2][1] = \begin{bmatrix} 1 & 0 & -2x\partial + 2x - 1 \end{bmatrix}$.

Таблица 1. Критерии сравнений строк $List2[1][2][1]$ и $List2[2][2][1]$

	Критерии сравнения	List2[1][2][1]	List2[2][2][1]
1	Максимальный порядок дифференцировния строки	3	1
2	Сумма порядков дифференцирования элементов строки	9	1
3	Количество степеней порядков диффренцирования	5	1
4	Суммарное количество слагаемых в коэффициентах дифференциальных операторов элементов строки	7	4
5	Сумма степеней полиномов в коэффициентах дифференциальных операторов элементов строки	5	2
6	Максимальный порядок полиномов в коэффициентах при ненулевых порядках дифференцирования	2	1
8	Сумма чисел по модулю при одночленах в коэффициентах дифференциальных операторов элементов строки	9	6
9	Сумма элементов строки	$(x-2)\partial^3+3\partial^2-(x^2+2x)\partial$	$-2x\partial+2x$

Очевидно, что характеристики 2-го столбца лучше характеристик 1-го, т. к. параметры меньше по значению. Значит, строка $List2[2][2][1]$ лучше строки $List2[1][2][1]$. Заметим, что в таблице параметр 9 можно сравнивать по критериям 1–8, однако правила следует применять к оператору в строгом порядке по возрастанию номеров критериев сравнения.

Аналогично сравним итоговые результирующие матрицы в списке *List2* (см. рисунок 2). Единственное различие состоит в том, что отличающиеся строки операторных матриц отличаются с точностью до постоянного множителя -1. Можно заметить, что одинаковые итоговые результирующие матрицы были получены за разное количество итераций.

Пример 9. Для большего понимания отличительных свойств результирующих операторных матриц, на вход программе подадим операторную матрицу (20), которая в ходе выполнения алгоритма Row-Reduction продемонстрирует связь одной неоднознач-

ности (см. п. 3.1.1) с другой (см. п. 3.1.2).

$$\begin{bmatrix} 5\partial^3 + (2-x)\partial^2 & 10x\partial^2 - x\partial & (-x^2-x)\partial \\ \partial^3 + 2\partial^2 + \partial & -x & x \\ 3\partial^2 & 1 & 1 \end{bmatrix} \tag{20}$$

Результат работы программы и разбор работы алгоритма RR над указанной матрицей приведён в Приложении А.

$$\begin{bmatrix} (3x-6)\partial^2 & -30x\partial^2 + (5+3x)\partial & (3x^2+3x+5)\partial \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \\ 3\partial^2 & 1 & 1 \end{bmatrix}, \begin{bmatrix} (3x-6)\partial^2 & -30x\partial^2 + (5+3x)\partial & (3x^2+3x+5)\partial \\ -6\partial^2 - 3\partial & \partial + 3x & \partial - 3x \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \end{bmatrix},$$

$$\begin{bmatrix} (8+x)\partial^2 + 5\partial & -10x\partial^2 + x\partial - 5x & (x^2+x)\partial + 5x \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \\ 3\partial^2 & 1 & 1 \end{bmatrix}, \begin{bmatrix} (8+x)\partial^2 + 5\partial & -10x\partial^2 + x\partial - 5x & (x^2+x)\partial + 5x \\ -6\partial^2 - 3\partial & \partial + 3x & \partial - 3x \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \end{bmatrix},$$

$$\begin{bmatrix} -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \\ (8+x)\partial^2 + 5\partial & -10x\partial^2 + x\partial - 5x & (x^2+x)\partial + 5x \\ 3\partial^2 & 1 & 1 \end{bmatrix}, \begin{bmatrix} (3x-6)\partial^2 & -30x\partial^2 + (5+3x)\partial & (3x^2+3x+5)\partial \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \\ 3\partial^2 & 1 & 1 \end{bmatrix},$$

$$\begin{bmatrix} (3x-6)\partial^2 & -30x\partial^2 + (5+3x)\partial & (3x^2+3x+5)\partial \\ (8+x)\partial^2 + 5\partial & -10x\partial^2 + x\partial - 5x & (x^2+x)\partial + 5x \\ -3\partial & \partial + 2 + 3x & \partial + 2 - 3x \end{bmatrix}$$

Рис. 3. Список операторных матриц каждой итерации выполнения алгоритма

Для операторной матрицы (20) все итоговые результирующие матрицы выглядят так, как показано на рисунке 3. Каждая из этих матриц в сравнении с любой другой матрицей имеет хотя бы одну общую строку. Если сравнивать две различающиеся строки между двумя матрицами, то можно воспользоваться критериями сравнения строк двух матриц из таблицы 1. Но этих критериев недостаточно, если количество строк больше одной, т. к. критерии сравнения в этом случае нестрогие.

Таким образом, исследовав достаточное количество выводов программы работы алгоритма RR, мы сформулировали критерии сравнения двух элементов операторной матрицы и двух строк, которые можно использовать для сравнения матриц и определять, какая из них обладает лучшими (меньшими по значению) характеритиками:

1. Максимальный порядок дифференцирования строки.

2. Сумма порядков дифференцирования элементов строки.

3. Количество степеней порядков диффренцирования.

4. Суммарное количество слагаемых в коэффициентах дифференциальных операторов элементов строки.

5. Сумма степеней полиномов в коэффициентах дифференциальных операторов элементов строки.

6. Максимальная степень полиномов в ненулевых коэффициентах дифференциальных операторов элементов строки.

7. Сумма чисел по модулю при одночленах в коэффициентах при порядках дифференцирования элементов строки.

8. Сумма элементов строки, к которой можно применить все вышеперечисленные критерии.

Такие критерии сравнения помогут в дальнейшем разрешить неоднозначный выбор пути работы алгоритма RR.

4.2 Операторные матрицы с хорошими характеристиками

Чтобы ввести понятие операторной матрицы, обладающей хорошими характеристиками, сначала решим, как лучше сравнивать результирующие матрицы.

Для этого рассмотрим сравнение двух результирующих матриц, у которых количество различающихся строк больше одной. Предварительно решим вопрос о сравнении двух строк операторных матриц, если критерии сравнения не дают нам однозначный результат.

4.2.1 Алгоритм наилучшего сопоставления элементов операторной матрицы между строками

Для решения вопроса приведем специальный алгоритм, на примере которого сравниваются две итоговые результирующие матрицы (см. рисунок 3), полученные в результате работы алгоритма над операторной матрицей (20):

$$\begin{bmatrix} (3x-6)\partial^2 & -30x\partial^2+(5+3x)\partial & (3x^2+3x+5)\partial \\ -6\partial^2-3\partial & \partial+3x & \partial-3x \\ -3\partial & \partial+2+3x & \partial+2-3x \end{bmatrix} \tag{21}$$

$$\begin{bmatrix} (8+x)\partial^2+5\partial & -10x\partial^2+x\partial-5x & (x^2+x)\partial+5x \\ -6\partial^2-3\partial & \partial+3x & \partial-3x \\ -3\partial & \partial+2+3x & \partial+2-3x \end{bmatrix} \tag{22}$$

Матрицы (21) и (22) отличаются первыми строками, которые мы сравним, не используя критерии сравнения строк п. 4.1. Необходимо сопоставить элементы строк по схожим характеристикам так, чтобы понять какая строка стала лучше другой.

Сформулируем критерии сравнения двух элементов операторной матрицы, являющихся дифференциальными операторами:

1. максимальный порядок дифференцирования элемента;
2. сумма порядков дифференцирования элемента;
3. количество степеней ненулевых порядков дифференцирования;
4. суммарное количество слагаемых в коэффициентах при порядках дифференцирования;
5. сумма степеней полиномов в коэффициентах при порядках дифференцирования;
6. максимальная степень полиномов в ненулевых коэффициентах при порядках дифференцирования;

Очевидно, что критерии сравнения двух элементов операторной матрицы почти не отличаются от критериев сравнения двух строк и выписаны в специальном порядке, чтобы в случае равенства параметров критерия i, мы могли бы продолжить сравнение с помощью критерия $i+1$.

Введём характеристику сравнения двух элементов *left* и *right* при сопоставлении строк операторных матриц (дадим характеристике обозначение «$cmpElems(left, right)$»):

1. «-1» — плохая характеристика у *left* по отношению к *right*;
2. «0» — схожие характеристики у *left* и у *right*;
3. «1» — хорошая характеристика у *left* по отношению к *right*.

Сопоставлением элементов двух строк или сопоставлением элементов строки Row_1 строке Row_2, называется вектор a, в котором каждый элемент показывает характеристику сравнения элемента строки Row_1 к элементу строки Row_2, т. е. число «-1», «0»

или «1», так, что для любого однозначного сопоставления элементов двух строк можно понять, насколько одна строка лучше другой. Один и тот же элемент строки Row_1 не может быть сопоставлен элементу другой строки Row_2 дважды.

Введем критерий сравнения сопоставлений элементов строк: вектор сопоставления a строки Row_1 лучше вектора сопоставления b строки Row_2, если

1. количество «-1» в векторе a меньше количества в векторе b, т. е. сравниваются плохие характеристики по условию «меньше», т. к. чем меньше сопоставлено элементов с плохим характеристиками строки Row_1 строке Row_2, тем лучше строка Row_1;

2. количество «1» в векторе a должно быть больше количества «1» в векторе b при условии, что количество «-1» в векторах одинаково, т. е. сравниваются хорошие характеристики по условию «больше», т. к чем больше сопоставлено элементов с хорошими характеристиками строки Row_1 строке Row_2, тем лучше строка Row_1.

Определив наилучшие критерии сравнения сопоставления строки Row_1 строке Row_2 по количеству полученных характеристик сравнения элементов строк, составим квадратную матрицу (см. таблицу 2) сравнений элементов строк $A = (a_{ij})$ размерности $m \times m$, где m — количество элементов в строке, где a_{ij} обозначает результат сравнения характеристик элементов $Row_1[i]$ и $Row_2[j]$.

Тогда, чтобы получить все вектора сопоставлений, необходимо заполнить *listVectors* всевозможными последовательностями вида (23). После чего будет получено $m!$ векторов всех сопоставлений элементов строк Row_1 и Row_2.

Алгоритм наилучшего сопоставления элементов операторной матрицы между строками

Вход: Две строки матриц размерности $m \times m$ Row_1, $Row_2 \in K[x][\partial]^{1\times m}$.

Выход: «$true$», если характеристики строки Row_1 лучше, чем характеристики строки Row_2, и «$false$» иначе.

Инициализация: $listVectors := []$.

Описание:

1. Составить матрицу $A = (a_{ij})$ размера $m \times m$, где каждый её элемент $a_{i,j}$ обозначает результат операции сравнения $Row_1[i]$ и $Row_2[j]$. $a_{i,j} \in \{-1, 0, 1\}$, $\forall i, j$ при $1 \leq i, j \leq m$.

2. Получить все возможные векторы из матрицы A сопоставления элементов строки Row_1 со строкой Row_2, и занести их в *listVectors*.

   Последовательность элементов матрицы, взятых по одному из каждой строки и каждого столбца:

   $$a_{1\alpha_1}, a_{2\alpha_2} \dots a_{m\alpha_m}. \tag{23}$$

   В последовательности элементы упорядочены по возрастанию номера строки, при этом номера столбцов $\alpha_1, \alpha_2, \dots, \alpha_m$ образуют перестановку из чисел $1, 2, \dots, m$, т. к. $\alpha_i \in \{1, 2, \dots, m\}$ и $\alpha_i \neq \alpha_j$ при $i \neq j$. Последовательностей вида (23) в матрице A столько, сколько существует перестановок $\alpha_1, \alpha_2, \dots, \alpha_m$ из m чисел, т. е. $m!$.

   Списком векторов сопоставления элементов двух строк *listVectors* квадратной матрицы $A = (a_{ij})$ m-го порядка называется набор всевозможных последовательностей $a_{1\alpha_1}, a_{2\alpha_2} \dots a_{m\alpha_m}$ элементов матрицы A, взятых по одному из каждой строки и каждого столбца. Каждая последовательность из m элементов образует вектор, который добавляется в *listVectors*.

3. Выбрать наилучший и наихудший векторы a и b сопоставления элементов строки Row_1 со строкой Row_2 из *listVectors* по критериям сравнения векторов сопоставления элементов строк.

4. Получить наилучший вектор сопоставления элементов строки Row_2 со строкой Row_1, равный произведению $-b$;

5. Сравнить наилучшие вектора a и $-b$ сопоставления элементов строк Row_1 и Row_2 соответственно по критериям сравнения векторов сопоставления элементов строк. Строка результирующей матрицы считается лучше другой, если соответствующий ей вектор сопоставления элементов строк является наилучшим:

   if вектор a лучше вектора $-b$ **then**

   return $true$;

   else

   return $false$;

   end if;

Пример 10. Для нашего примера строки Row_1 и Row_2 из матриц (21) и (22) имеют вид:

$$Row_1 = \Big[(3x-6)\partial^2 \quad -30x\partial^2 + (5+3x)\partial \quad (3x^2+3x+5)\partial\Big],$$

$$Row_2 = \Big[(8+x)\partial^2 + 5\partial \quad -10x\partial^2 + x\partial - 5x \quad (x^2+x)\partial + 5x\Big]$$

Таблица 2. Матрица $A = (a_{ij})$ сравнений элементов строк Row_1 и Row_2

Row_1&Row_2	$(8+x)\partial^2 + 5\partial$	$-10x\partial^2 + x\partial - 5x$	$(x^2+x)\partial + 5x$
$(3x-6)\partial^2$	1	1	-1
$-30x\partial^2 + (5+3x)\partial$	-1	1	-1
$(3x^2+3x+5)\partial$	1	1	1

Выпишем все возможные векторы (комбинации) сопоставления элементов строки Row_1 строке Row_2 по вычисленным характеристикам сравнения двух элементов:

$$\Big[1 \quad 1 \quad 1\Big], \Big[1 \quad 1 \quad -1\Big], \Big[-1 \quad 1 \quad 1\Big], \Big[-1 \quad 1 \quad -1\Big], \Big[1 \quad 1 \quad -1\Big], \Big[1 \quad 1 \quad -1\Big].$$

Каждая комбинация была получена следующим образом:

1. $cmpElems(Row_1[1], Row_2[1]) = 1$, $cmpElems(Row_1[2], Row_2[2]) = 1$ и $cmpElems(Row_1[3], Row_2[3]) = 1$;

2. $cmpElems(Row_1[1], Row_2[1]) = 1$, $cmpElems(Row_1[3], Row_2[2]) = 1$ и $cmpElems(Row_1[2], Row_2[3]) = -1$;

3. $cmpElems(Row_1[2], Row_2[1]) = -1$, $cmpElems(Row_1[1], Row_2[2]) = 1$ и $cmpElems(Row_1[3], Row_2[3]) = 1$;

4. $cmpElems(Row_1[2], Row_2[1]) = -1$, $cmpElems(Row_1[3], Row_2[2]) = 1$ и $cmpElems(Row_1[1], Row_2[3]) = -1$;

5. $cmpElems(Row_1[3], Row_2[1]) = 1$, $cmpElems(Row_1[1], Row_2[2]) = 1$ и $cmpElems(Row_1[2], Row_2[1]) = -1$;

6. $cmpElems(Row_1[3], Row_2[1]) = 1$, $cmpElems(Row_1[2], Row_2[1]) = 1$ и $cmpElems(Row_1[1], Row_2[2]) = -1$.

Заметим, что получились две одинаковые комбинации $\begin{bmatrix}1 & 1 & -1\end{bmatrix}$, т. к. возможны два разных сопоставления с одинаковыми характеристиками сравнения элементов.

Самая лучшая комбинация — $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$, т. к. характеристики сравнения сопоставленных элементов строк Row_1 и Row_2 имеют вид: $cmpElems(Row_1[1], Row_2[1]) = 1$, $cmpElems(Row_1[2], Row_2[2]) = 1$ и $cmpElems(Row_1[3], Row_2[3]) = 1$.

Кроме поиска лучшей комбинации сопоставления элементов строки Row_1 строке Row_2 необходимо найти лучшую комбинацию сопоставления элементов строки Row_2 строке Row_1, чтобы сравнить эти комбинации и определить, какая строка результирующей матрицы лучше другой.

Очевидно, что при сравнении строки Row_2 со сторкой Row_1, наихудшее сопоставление строки Row_1 со сторкой Row_2 — это есть наилучшее сопоставление при сравнении строки Row_2 со сторкой Row_1.

Наихудшая комбинация сопоставления строки Row_1: $\begin{bmatrix}-1 & 1 & -1\end{bmatrix}$, значит, наилучшей комбинацией сопоставления строки Row_2 будет вектор $\begin{bmatrix}1 & -1 & 1\end{bmatrix} = -\begin{bmatrix}-1 & 1 & -1\end{bmatrix}$.

Чтобы сравнить лучшие комбинаций строк Row_1 и Row_2 и выяснить, какая строка лучше другой следует:

1. найти наилучший и наихудший векторы сопоставления сравнений строки Row_1 со сторкой Row_2;

2. наихудший вектор сопоставления умножить на -1 и сравнить с лучшим.

Строка Row_1 является лучше строки Row_2, если по критерию сравнения сопоставлений элементов строк вектор сопоставления строки Row_1 лучше вектора сопоставления строки Row_2.

Лучшим вектором сопоставления элементов строк Row_1 и Row_2 среди векторов $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$ и $\begin{bmatrix}1 & -1 & 1\end{bmatrix}$ соответственно будет вектор $\begin{bmatrix}1 & 1 & 1\end{bmatrix}$, следовательно, строка Row_1 лучше строки Row_2.

Сравнение лучших комбинаций строк Row_1 и Row_2 позволит определить результирующую матрицу с хорошими характеристиками, имеющую строку с лучшей комбинацией сопоставления элементов строк Row_1 и Row_2.

Следовательно, результирующая матрица (21) лучше матрицы (22), т. к. строка Row_1 обладает лучшими характеристиками, чем строка Row_2.

Таким образом, мы можем качественно сравнивать строки результирующих матриц, если различающихся строк не более одной.

4.2.2 Алгоритм наилучшего сопоставления строк между матрицами

Для сравнивнения строк результирующих матриц, у которых количество различных строк больше одной, воспользуемся той же схемой, которая была представлена в алгоритме наилучшего сопоставления элементов строк операторной матрицы п. 4.2.1. Сопоставим различающиеся строки результирующих матриц так, чтобы определить какая результирующая матрица лучше.

Если существуют две результирующие матрицы M_1 и M_2 размерности $m \times m$ вида: $\begin{bmatrix} M_1[1] \\ M_1[2] \\ \dots \\ M_1[m] \end{bmatrix}$, $\begin{bmatrix} M_2[1] \\ M_2[2] \\ \dots \\ M_2[m] \end{bmatrix}$, где $M_i[j] = Row_j(M_i)$ при $\forall i \in \{1, 2\}$ и $1 \leq j \leq m$, у которых количество различающихся строк больше одной, то возможно сравнение строк по критериям сравнения (см. п. 4.1) или по алгоритму наилучшего сопоставления элементов строк операторной матрицы (см. п. 4.2.1). Данные правила будут использованы для сравнения строк в алгоритме далее.

Алгоритм наилучшего сопоставления строк между матрицами

Вход: Различные строки результирующих матриц M_1 и M_2 размера $m \times m$: $Row_{k_i}(M_1) \neq Row_{k_j}(M_2)$, где строки $Row_{k_i}(M_1)$, $Row_{k_j}(M_2) \in K[x][\partial]^{1 \times m}$ при $1 \leq k_i, k_j \leq n$ и $1 \leq i, j \leq m$, где n — количество различных строк матриц M_1 и M_2.

Выход: «$true$», если характеристики матрицы M_1 лучше, чем характеристики матрицы M_2, и «$false$» иначе.

Инициализация: $listRowVectors := []$.

Описание:

1. Если $n \neq 1$, то перейти к следующему шагу, иначе сравнить две строки результирующих матриц по правилам:
   (a) Критерии сравнения строк (см. п. 4.1).
   (b) Алгоритм наилучшего сопоставления элементов строк операторных матриц.
2. Составить матрицу $B = (b_{ij})$ размера $n \times n$, где каждый элемент b_{ij} обозначает результат операции сравнения $Row_{k_i}(M_1)$ и $Row_{k_j}(M_2)$. $b_{ij} \in \{-1, 0, 1\}$, $1 \leq i, j \leq n$ при $1 \leq k \leq m$.

3. Получить все возможные векторы из матрицы B размера $n \times n$ сопоставления строк матрицы M_1 с матрицей M_2, и добавить их в *listRowVectors*.

   Составить последовательность элементов матрицы, взятых по одному из каждой строки и каждого столбца:

   $$b_{1\beta_1}, b_{2\beta_2} \dots b_{n\beta_n}. \tag{24}$$

   В последовательности элементы упорядочены по возрастанию номера строки, при этом номера столбцов $\beta_1, \beta_2, \dots, \beta_n$ образуют перестановку из чисел $1, 2, \dots, n$, т. к. $\beta_i \in \{1, 2, \dots, n\}$ и $\beta_i \neq \beta_j$ при $i \neq j$. Последовательностей вида (24) в матрице B столько, сколько существует перестановок $\beta_1, \beta_2, \dots, \beta_n$ из n чисел, т. е. $n!$.

   Списком векторов сопоставления строк двух матриц *listRowVectors* квадратной матрицы B размера $n \times n$ называется набор всевозможных последовательностей $b_{1\beta_1}, b_{2\beta_2} \dots b_{n\beta_n}$ элементов матрицы B, взятых по одному из каждой строки и каждого столбца. Каждая последовательность из n элементов образует вектор, который добавляется в *listRowVectors*.

4. Выбрать наилучший и наихудший векторы ν_1 и ν_2 сопоставления строк матрицы M_1 с матрицей M_2 из *listRowVectors* по критериям сравнения векторов сопоставления строк матриц.

5. Получить наилучший вектор сопоставления строк матрицы M_2 с матрицей M_1, равный произведению $-\nu_2$.

6. Сравнить наилучшие вектора ν_1 и $-\nu_2$ сопоставления строк матриц M_1 и M_2 соответственно по критериям сравнения векторов сопоставления строк матриц. Результирующая матрица считается лучше, если соответствующий ей вектор сопоставления строк матриц является наилучшим:

   if вектор ν_1 лучше вектора $-\nu_2$ **then**
   return $true$;
   else
   return $false$;
   end if;

Введём характеристику сравнения двух строк результирующих матриц *leftRow* и *rightRow* при сопоставлении строк операторных матриц (дадим характеристике обозначение «*cmpRows(leftRow,rightRow)*»):

1. «-1» — плохая характеристика у *leftRow* по отношению к *rightRow*;

2. «0» — схожие характеристики у *leftRow* и у *rightRow*;

3. «1» — хорошая характеристика у *leftRow* по отношению к *rightRow*.

Сопоставлением строк двух результирующих матриц или сопоставлением строк матрицы M_1 матрице M_2 называется вектор a, в котором каждый элемент показывает характеристику сравнения строки матрицы M_1 со строкой матрицы M_2, т. е. число «-1», «0» или «1», так, что для любого однозначного сопоставления строк двух матриц можно понять, насколько одна матрица лучше другой. Одна и та же строка матрицы M_1 не может быть сопоставлена другой матрице M_2 дважды.

Введем критерий сравнения сопоставлений строк матриц: вектор сопоставления a матрицы M_1 будем называть лучше вектора сопоставления b матрицы M_2, если:

1. количество «-1» в векторе a меньше количества в векторе b, т. е. сравниваются плохие характеристики по условию «меньше», т. к. чем меньше сопоставлено строк с плохим характеристиками матрицы M_1 матрице M_2, тем лучше матрица M_1;

2. количество «1» в векторе a больше количества «1» в векторе b при условии, что количество «-1» в векторах одинаково. Сравниваются хорошие характеристики по условию «больше», т. к. чем больше сопоставлено строк с хорошими характеристиками матрицы M_1 матрице M_2, тем лучше матрица M_1.

В виде таблицы 3 составлена квадратная матрица сравнений строк матриц $B = (b_{ij})$ размерности $n \times n$, где n — количество различных строк матриц, b_{ij} обозначает результат сравнения характеристик различных строк $M_1[i]$ и $M_2[j]$, т. е. $b_{ij} \in \{-1, 0, 1\}$.

Для получения всех векторов сопоставлений различных строк матриц, необходимо заполнить *listRowVectors* всевозможными последовательностями вида (24). После чего будет получено $n!$ векторов всех сопоставлений строк матрицы M_1 и M_2.

Пример 11. На примере операторной матрицы (12) показано, как сравнить две итоговые результирующие матрицы по строкам.

Ниже представлены две итоговые результирующие матрицы:

$$M_1 = \begin{bmatrix} -9\partial^2 - 3\partial & -33\partial^2 + 7x & -6x^2 + 4 & 4 \\ x\partial^2 + x & (x^2 + x)\partial & 6\partial^2 & 1 \\ -3\partial^2 + 33\partial & 7x\partial + 14x + 7 & (-6x^2 + 4)\partial - 12x + 14 & 4\partial + 14 \\ 33\partial^2 & 7x\partial^2 + (14x + 14)\partial + x + 14 & (-6x^2 + 4)\partial^2 - (24x - 14)\partial - 11 & 4\partial^2 + 14\partial + 1 \end{bmatrix},$$

$$M_2 = \begin{bmatrix} 240\partial^2 + 3\partial & (49x + 33)\partial^2 + (98x + 98)\partial + 98 & (28 - 42x^2)\partial^2 - (168x - 98)\partial + 6x^2 - 81 & 28\partial^2 + 98\partial + 3 \\ x\partial^2 + x & (x^2 + x)\partial & 6\partial^2 & 1 \\ 6\partial^2 + 240\partial & 99\partial^2 + 49x\partial + 77x + 49 & (28 - 42x^2)\partial + 18x^2 - 84x + 86 & 28\partial + 86 \\ -9\partial^2 - 3\partial & -33\partial^2 + 7x & -6x^2 + 4 & 4 \end{bmatrix}.$$

Матрицы M_1 и M_2 отличаются двумя строками: третья и четвёртая строки из M_1, и первая и третья строки из M_2. Для того чтобы понять, какая результирующая матрица лучше, сопоставлены строки матриц в таблице 3.

Таблица 3. Матрица $B = (b_{ij})$ сравнений строк матриц M_1 и M_2

$Rows_{M_1}\&Rows_{M_2}$	$Row_1(M_2)$	$Row_3(M_2)$
$Row_3(M_1)$	1	1
$Row_4(M_1)$	1	-1

Выпишем все возможные векторы (комбинации) сопоставления строк матрицы M_1 матрице M_2 по вычисленным характеристикам сравнения двух строк:

$$\begin{bmatrix} 1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

Каждая комбинация получена следующим образом:

1. $cmpRows(Row_3(M_1), Row_1(M_2)) = 1$ и $cmpRows(Row_4(M_1), Row_3(M_2)) = -1$;
2. $cmpRows(Row_4(M_1), Row_1(M_2)) = 1$ и $cmpRows(Row_3(M_1), Row_3(M_2)) = 1$.

Самая лучшая комбинация для матрицы M_1 — $\begin{bmatrix} 1 & 1 \end{bmatrix}$.

Кроме поиска лучшей комбинации сопоставления строк матрицы M_1 матрице M_2 необходимо найти лучшую комбинацию сопоставления строк матрицы M_2 матрице M_1, чтобы сравнить эти комбинации и определить, какая результирующая матрица лучше.

Очевидно, что при сравнении M_2 с M_1, наихудшее сопоставление M_1 с M_2 — есть наилучшее сопоставление при сравнении M_2 с M_1.

Наихудшая комбинация сопоставления матрицы M_1 с матрицей M_2: $\begin{bmatrix} 1 & -1 \end{bmatrix}$, значит, наилучшей комбинацией сопоставления для M_2 будет вектор $\begin{bmatrix} -1 & 1 \end{bmatrix} = -\begin{bmatrix} 1 & -1 \end{bmatrix}$.

Чтобы сравнить лучшие комбинаций матриц M_1 и M_2 и выяснить, какая из матриц лучше, следует:

1. найти наилучший и наихудший векторы сопоставления сравнений матрицы M_1 с матрицей M_2;
2. наихудший вектор сопоставления умножить на -1 и сравнить с лучшим для M_1.

Лучшим вектором сопоставления строк матриц M_1 и M_2 среди векторов $\begin{bmatrix}1 & 1\end{bmatrix}$ и $\begin{bmatrix}-1 & 1\end{bmatrix}$ соответственно будет вектор $\begin{bmatrix}1 & 1\end{bmatrix}$, следовательно, матрица M_1 лучше матрицы M_2.

Следовательно, результирующая матрица M_1 лучше матрицы M_2, т. к. по критерию сравнения сопоставлений строк матриц вектор сопоставления матрицы M_1 лучше вектора сопоставления матрицы M_2.

Таким образом, мы можем качественно сравнивать строки результирующих матриц, чтобы определить, какая матрица обладает лучшими характеристиками.

4.2.3 Понятие матрицы с хорошими характеристиками

Сформулировав критерии сравнения результирующих операторных матриц, можно определить операторную матрицу лучшего вида.

Под лучшим видом операторной матрицы понимается результирующая матрица, вид которой тем приятнее, чем меньше памяти необходимо для её представления в памяти ЭВМ, т. е. будут меньше:

1. степени полиномов;
2. порядки дифференцирования;
3. число слагаемых в полиномах.

Более простой вид матриц уменьшает вычислительную сложность их последующей обработки.

Поэтому, следует выстроить последовательный, строгий порядок обхода критериев сравнения и эвристических правил, направленных на разрешение неоднозначного выбора в работе алгоритма RR с меньшими затратами ресурсов ЭВМ, чтобы всегда ясно определять результат сравнения матриц. По данному списку следует двигаться последовательно до получения однозначного результата, когда характеристики станут разными.

4.3 Итоговые эвристические правила

Проанализировав вид всех возможных результирующих матриц с целью выработки критериев сравнения двух матриц, определив понятие операторной матрицы, обладающей хорошими характеристиками, сформулируем эвристические правила, позволяющие

разрешать неоднозначности алгоритма RR, чтобы получить результирующую матрицу с хорошими характеристиками.

Во время обработки неоднозначностей алгоритма RR был проведен локальный анализ новых результирующих матриц, т. е. на момент текущей итерации алгоритма. При этом достаточно сравнить различные строки результирующих матриц по алгоритму наилучшего сопоставления строк между матрицами (см. п. 4.2.2), включая строки, полученные на шаге 3 алгоритма RR. Эксперименты показали, что при наличии неоднозначностей на текущей итерации алгоритма RR результирующие матрицы отличаются одной строкой с учётом перестановок строк.

Эвристические правила сравнения элементов операторной матрицы:

1. Максимальный порядок дифференцирования элемента.
2. Сумма порядков дифференцирования элемента.
3. Количество степеней ненулевых порядков дифференцирования.
4. Суммарное количество слагаемых в коэффициентах порядков дифференцирования.
5. Сумма порядков полиномов в коэффициентах порядков дифференцирования.
6. Максимальный порядок полиномов в коэффициентах ненулевых порядков дифференцирования.

Эвристические правила сравнения строк операторной матрицы:

1. Максимальный порядок дифференцировния строки.
2. Сумма порядков дифференцирования элементов строки.
3. Количество степеней порядков диффренцирования.
4. Суммарное количество слагаемых в коэффициентах порядков дифференцирования.
5. Сумма порядков полиномов в коэффициентах порядков дифференцирования.
6. Максимальный порядок полиномов в коэффициентах ненулевых порядков дифференцирования.

7. Максимальный порядок полиномов в коэффициентах ненулевых порядков дифференцирования.

8. Сумма элементов строки, к которой нужно применить эвристические правила сравнения элементов операторной матрицы.

9. К сравниваемым строкам применить алгоритм наилучшего сопоставления элементов операторной матрицы между строками.

Эвристические правила разрешения недетерминированности алгоритма RR:

1. Для выбора вектора коэффициентов линейной зависимости строк фронтальной матрицы заменив новую строку на шаге 3 алгоритма RR, применить алгоритм наилучшего сопоставления строк между полученными результирующими матрицами.

2. Для выбора элемента вектора коэффициентов линейной зависимости строк фронтальной матрицы мы сравниваем строки, номера которых определяются на шаге 2 алгоритма RR, исходной операторной матрицы по строго определённым эвристическим правилам сравнения строк.

5 Реализация улучшенной версии алгоритма Row-Reduction

Все реализации работают только с квадратными операторными матрицами полного ранга. Специальная версия RR, использующая предложенные правила для разрешения неоднозначного выбора в ходе работы алгоритма, была реализована в системе компьютерной алгебры Maple в виде процедуры *modifyRR*.

Данная реализация опирается на возможности пакетов *OreTools*, *LinearAlgebra*, *RandomTools*. Пакет *LinearAlgebra* использовался с целью обращения к процедурам:

1. *LinearAlgebra[NullSpace]* для вычисления коэффициентов линейной зависимости строк фронтальной матрицы.

2. *LinearAlgebra[Rank]* для вычисления ранга фронтальной матрицы в качестве условия проверки окончания работы алгоритма RR (2).

3. *LinearAlgebra[RandomMatrix]* для генерации матриц.

Дифференциальные операторы задаются с помощью специальных структур *OrePoly*, предоставляемых пакетом *OreTools* и соответствующих внутреннему представлению некоммутативных полиномов Ope. Кроме того, из пакета OreTools были задействованы процедуры Add и Multiply для преобразования строки операторной матрицы на шаге 3 алгоритма RR. Пакет *RandomTools* использовался для генерации чисел и полиномов с помощью процедуры *Generate* и стандартной процедуры *randpoly*.

На вход процедуры *modifyRR* поступает произвольная операторная матрица. При наличии неоднозначностей на каждой итерации программа проводит локальный анализ результирующих матриц, соблюдая строгий порядок выполнения эвристических правил. Алгоритм завершается, когда итоговая результирующая матрица приведена по строкам, и тогда программа возвращает итоговую результирующую матрицу.

Процедура *modifyRR* имеет следующие входные параметры:

modifyRR(opMatrix::Matrix)

opMatrix — операторная матрица.

Результатом работы программы является итоговая результирующая матрица.

Приведём примеры для полного понимания работы этой программы с добавочной печатью информации о результирующих матрицах, вычисленных на каждой итерации

алгоритма, и пр. Из-за того, что в системе компьютерной алгебре Maple символ ∂ зарезервирован, поэтому при выводе операторных матриц дифференциальный оператор будет обозначаться, как δ.

Пример 12. Подаём на вход программе операторную матрицу:

$$> Matrix_1 := \begin{bmatrix} OrePoly(7) & OrePoly(12,3,0,0,5-11x) & OrePoly(-3,-6) \\ OrePoly(-10) & OrePoly(7,12+9x) & OrePoly(-13) \\ OrePoly(5) & OrePoly(-9,0,0,-7-2x) & OrePoly(11,0,-4) \end{bmatrix}$$

$> modifyRR(Matrix_1)$

Её фронтальная матрица вырождена и имеет два вектора коэффициентов линейной зависимости:

$$\begin{bmatrix} -2x-7 \\ 0 \\ 11x-5 \end{bmatrix}, \begin{bmatrix} 9x+12 \\ 11x-5 \\ 0 \end{bmatrix}$$

Возникла неоднозначность выбора вектора. Замена новой строки будет совершена по первой строке по каждому вектору коэффициентов линейной зависимости строк фронтальной матрицы. Результирующие матрицы Mat_1 и Mat_2 соответственно:

$$\begin{bmatrix} (55x-25)\delta - 14x - 49 & (-22x+10)\delta^3 + (-105x+24)\delta - 24x - 84 & (-44x+20)\delta^3 + (133x-13)\delta + 6x + 21 \\ -10 & (9x+12)\delta + 7 & -13 \\ 5 & (-2x-7)\delta^3 - 9 & -4\delta^2 + 11 \end{bmatrix}$$

$$\begin{bmatrix} (-110x+50)\delta^3 + 63x + 84 & (374x-170)\delta^3 + (27x+36)\delta + 108x + 144 & (-143x+65)\delta^3 + (-54x-72)\delta - 27x - 36 \\ -10 & (9x+12)\delta + 7 & -13 \\ 5 & (-2x-7)\delta^3 - 9 & -4\delta^2 + 11 \end{bmatrix}$$

Mat_1 и Mat_2 различаются первыми строками. Первая строка Mat_1 лучше первой строки Mat_2 по эвристическому правилу — сумма порядков дифференцирования элементов строк. Значит, Mat_1 лучше Mat_2.

Для Mat_1 фронтальная матрица вырождена, которая имеет один линейно независимый вектор линейной зависимости, у которого нет неоднозначности выбора строки операторной матрицы. Вектор линейной зависимости строк фронтальной матрицы результирующей матрицы Mat_1:

$$\begin{bmatrix} 0 \\ 2x+7 \\ 9x+12 \end{bmatrix}$$

Вычислим по этому вектору очередную результирующую матрицу, у которой фронтальная матрица невырождена:

$$\begin{bmatrix} (55x-25)\delta-14x-49 & (-22x+10)\delta^3+(-105x+24)\delta-24x-84 & (-44x+20)\delta^3+(133x-13)\delta+6x+21 \\ -10 & (9x+12)\delta+7 & -13 \\ (-20x-70)\delta^2+45x+60 & (50x+175)\delta^2-81x-108 & (-62x-139)\delta^2+99x+132 \end{bmatrix}$$

Пример 13. Подадим на вход программе новую операторную матрицу:

$$> Matrix_2 := \begin{bmatrix} OrePoly(2,0,29-x) & OrePoly(29) & OrePoly(5,23) \\ OrePoly(6,28+21x) & OrePoly(0) & OrePoly(0) \\ OrePoly(-26,-19,-1+17x) & OrePoly(0,-2) & OrePoly(4,-29) \end{bmatrix}$$

$> modifyRR(Matrix_2)$

Её фронтальная матрица вырождена и имеет два вектора ν_1 и ν_2 коэффициентов линейной зависимости:

$$\begin{bmatrix} 17x-1 \\ 0 \\ x-29 \end{bmatrix}, \begin{bmatrix} 21x+28 \\ x-29 \\ 0 \end{bmatrix}$$

Из-за того, что вектор порядков дифференцирования строк операторной матрицы равен:

$$\text{"m\_deg\_rows"}, \begin{bmatrix} 2 & 1 & 2 \end{bmatrix}$$

поэтому, для первого вектора коэффициентов линейной зависимости строк фронтальной матрицы существует неоднозначный выбор строки операторной матрицы, первой и третьей, с одинаковым порядком дифференцирования строки, равным 2. Тогда для локального анализа 1-й итерации исследуем три результирующие матрицы $Mat_{\nu_1,Row_3(Matrix_2)}$ и $Mat_{\nu_1,Row_3(Matrix_2)}$, первая и вторая из которых получились по первому вектору ν_1 коэффициентов линейной зависимости строк фронтальной матрицы с учётом выбора номера элемента ν_1, а Mat_{ν_2} третья матрица — по второму вектору ν_2:

$$\begin{bmatrix} (-19x+551)\delta+8x+752 & (-2x+58)\delta+493x-29 & (362x+818)\delta+89x-121 \\ (21x+28)\delta+6 & 0 & 0 \\ (17x-1)\delta^2-19\delta-26 & -2\delta & -29\delta+4 \end{bmatrix}$$

$$\begin{bmatrix} (-x+29)\delta^2+2 & 29 & 23\delta+5 \\ (21x+28)\delta+6 & 0 & 0 \\ (-19x+551)\delta+8x+752 & (-2x+58)\delta+493x-29 & (362x+818)\delta+89x-121 \end{bmatrix}$$

$$\begin{bmatrix} (27x-783)\delta+42x+56 & 609x+812 & (483x+644)\delta+105x+140 \\ (21x+28)\delta+6 & 0 & 0 \\ (17x-1)\delta^2-19\delta-26 & -2\delta & -29\delta+4 \end{bmatrix}$$

$Mat_{\nu_1,Row_3(Matrix_2)}$ лучше $Mat_{\nu_1,Row_1(Matrix_2)}$, т. к. первая строка $Mat_{\nu_1,Row_3(Matrix_2)}$ лучше третьей строки $Mat_{\nu_1,Row_1(Matrix_2)}$ вторые строки различаются, и строка Mat_{ν_1} лучше строки Mat_{ν_1} по эвристическому правилу — сумма максимальных порядков дифференцирования. Для сравнения $Mat_{\nu_1,Row_3(Matrix_2)}$ и Mat_{ν_2} воспользуемся алгоритмом наилучшего сопоставления строк между матрицами (см. п. 4.2.2). Построим таблицу сопоставления строк между матрицами $Mat_{\nu_1,Row_3(Matrix_2)}$ и Mat_{ν_2}

$$\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

и можно заметить, что при наилучшем сопоставлении первой строки $Mat_{\nu_1,Row_3(Matrix_2)}$ с первой строкой Mat_{ν_2} и третьей - с третьей. Следовательно, $Mat_{\nu_1,Row_3(Matrix_2)}$ лучше Mat_{ν_2}.

$$\begin{bmatrix} (-x+29)\,\delta^2+2 & 29 & 23\,\delta+5 \\ (21\,x+28)\,\delta+6 & 0 & 0 \\ (-19\,x+551)\,\delta+8\,x+752 & (-2\,x+58)\,\delta+493\,x-29 & (362\,x+818)\,\delta+89\,x-121 \end{bmatrix}$$

Результирующая матрица $Mat_{\nu_1,Row_3(Matrix_2)}$ имеет вырожденную фронтальную матрицу, у которой один вектор коэффициентов линейной зависимости, по которому нет неоднозначности выбора строки операторной матрицы. После чего получим результирующую итоговую матрицу, полученную в результате работы процедуры *modifyRR*:

$$\begin{bmatrix} (27\,x-783)\,\delta+42\,x+56 & 609\,x+812 & (483\,x+644)\,\delta+105\,x+140 \\ (21\,x+28)\,\delta+6 & 0 & 0 \\ (-19\,x+551)\,\delta+8\,x+752 & (-2\,x+58)\,\delta+493\,x-29 & (362\,x+818)\,\delta+89\,x-121 \end{bmatrix}$$

Программа показала, как разрешает неоднозначный выбор работы алгоритма с помощью эвристических правил. Это было видно в примере 13 по 1-й итерации алгоритма, где было три возможных развития работы программы.

Для проверки работы специальной версии программы с эвристическими правилами будем сравнивать две итоговые результирующие матрицы, одна из которых была получена в результате работы процедуры *modifyRR*, а вторая матрица является лучшей среди всех итоговых результирующих матриц, которая была получена в результате сравнения матриц по эвристическим правилам.

Для нахождения лучшей матрицы среди итоговых результирующих матриц была разработана процедура *getBetterResultMatrix*, которая по эвристическим правилам выбирает наилучшую. На вход ей подаётся список итоговых результирующих матриц и

наименьшее количество итераций, за которое алгоритм завершил работу. Эти параметры были получены по глобальным параметрам в результате работы программы *outputRR*, собирающей в единую структуру все результирующие матрицы (см. п. 4.1), во время работы которой глобальные параметры собирают информацию о итоговых результирующих матрицах и за сколько итераций они были получены.

По примерам 12 и 13 найдём все итоговые результирующие матрицы, среди которых найдём лучшую, вычисленную с помощью процедуры *getBetterResultMatrix*. Сравним вычисленные матрицы с матрицами, полученными в результате работы процедуры *modifyRR*.

Пример 14. В примере 12 на вход программе *modifyRR* была подана операторная матрица,

$$> \ Matrix_1 := \begin{bmatrix} OrePoly(7) & OrePoly(12,3,0,0,5-11x) & OrePoly(-3,-6) \\ OrePoly(-10) & OrePoly(7,12+9x) & OrePoly(-13) \\ OrePoly(5) & OrePoly(-9,0,0,-7-2x) & OrePoly(11,0,-4) \end{bmatrix}$$

$> \ modifyRR(Matrix_1)$

в результате чего была получена итоговая результирующая *modifyMat*:

$$\begin{bmatrix} (55x-25)\,\delta-14x-49 & (-22x+10)\,\delta^3+(-105x+24)\,\delta-24x-84 & (-44x+20)\,\delta^3+(133x-13)\,\delta+6x+21 \\ -10 & (9x+12)\,\delta+7 & -13 \\ (-20x-70)\,\delta^2+45x+60 & (50x+175)\,\delta^2-81x-108 & (-62x-139)\,\delta^2+99x+132 \end{bmatrix}$$

Для вычисления лучшей матрицы *betMat* среди итоговых результирующих

$$\begin{bmatrix} (55x-25)\,\delta-14x-49 & (-22x+10)\,\delta^3+(-105x+24)\,\delta-24x-84 & (-44x+20)\,\delta^3+(133x-13)\,\delta+6x+21 \\ -10 & (9x+12)\,\delta+7 & -13 \\ (-20x-70)\,\delta^2+45x+60 & (50x+175)\,\delta^2-81x-108 & (-62x-139)\,\delta^2+99x+132 \end{bmatrix}$$

$$\begin{bmatrix} (-110x+50)\,\delta^3+63x+84 & (374x-170)\,\delta^3+(27x+36)\,\delta+108x+144 & (-143x+65)\,\delta^3+(-54x-72)\,\delta-27x-36 \\ -10 & (9x+12)\,\delta+7 & -13 \\ (-20x-70)\,\delta^2+45x+60 & (50x+175)\,\delta^2-81x-108 & (-62x-139)\,\delta^2+99x+132 \end{bmatrix}$$

воспользуемся процедурой *getBetterResultMatrix*:

$> \ getBetterResultMatrix(UID\_Results,\ UID\_iteration)$

$$\begin{bmatrix} (55x-25)\,\delta-14x-49 & (-22x+10)\,\delta^3+(-105x+24)\,\delta-24x-84 & (-44x+20)\,\delta^3+(133x-13)\,\delta+6x+21 \\ -10 & (9x+12)\,\delta+7 & -13 \\ (-20x-70)\,\delta^2+45x+60 & (50x+175)\,\delta^2-81x-108 & (-62x-139)\,\delta^2+99x+132 \end{bmatrix}$$

Матрицы *betMat* и *modifyMat* совпадают.

Пример 15. В результате работы программы *outputRR*, на вход которой была подана операторная матрица из примера 13,

$$> Matrix_2 := \begin{bmatrix} OrePoly(2, 0, 29 - x) & OrePoly(29) & OrePoly(5, 23) \\ OrePoly(6, 28 + 21x) & OrePoly(0) & OrePoly(0) \\ OrePoly(-26, -19, -1 + 17x) & OrePoly(0, -2) & OrePoly(4, -29) \end{bmatrix}$$

$> outputRR(Matrix_2)$

мы собрали информацию об итоговых результирующих матрицах. Все её итоговые результирующие матрицы выполнились за одинаковое количество итераций, равное двум. Также для наглядности можно показать в виде дерева, графа, как работал алгоритм на каждой итерации, где корнем дерева является операторная матрица:

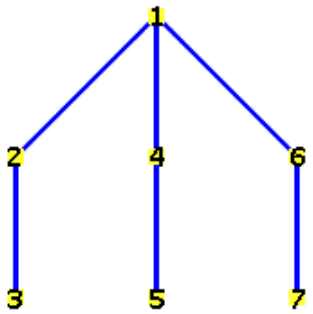
"End Results"

$$\begin{bmatrix} OrePoly(752 + 8x, -19x + 551) & OrePoly(-29 + 493x, -2x + 58) & OrePoly(-121 + 89x, 362x + 818) \\ OrePoly(6, 21x + 28) & OrePoly(0) & OrePoly(0) \\ OrePoly(-728 - 546x, -858x - 505) & OrePoly(0, -42x - 56) & OrePoly(112 + 84x, -609x - 812) \end{bmatrix}$$

$$\begin{bmatrix} OrePoly(42x + 56, 27x - 783) & OrePoly(609x + 812) & OrePoly(105x + 140, 483x + 644) \\ OrePoly(6, 21x + 28) & OrePoly(0) & OrePoly(0) \\ OrePoly(752 + 8x, -19x + 551) & OrePoly(-29 + 493x, -2x + 58) & OrePoly(-121 + 89x, 362x + 818) \end{bmatrix}$$

$$\begin{bmatrix} OrePoly(42x + 56, 27x - 783) & OrePoly(609x + 812) & OrePoly(105x + 140, 483x + 644) \\ OrePoly(6, 21x + 28) & OrePoly(0) & OrePoly(0) \\ OrePoly(-728 - 546x, -858x - 505) & OrePoly(0, -42x - 56) & OrePoly(112 + 84x, -609x - 812) \end{bmatrix}$$

"numberIteration=", 2



В результате работы процедуры *getBetterResultMatrix* получим лучшую матрицу среди итоговых результирующих.

$> betMat := getBetterResultMatrix(UID\_Results, UID\_iteration)$

Эта матрица отличается от той, что была получена в результате работы *modifyRR*:

$$\begin{bmatrix} (27x-783)\,\delta+42x+56 & 609x+812 & (483x+644)\,\delta+105x+140 \\ (21x+28)\,\delta+6 & 0 & 0 \\ (-19x+551)\,\delta+8x+752 & (-2x+58)\,\delta+493x-29 & (362x+818)\,\delta+89x-121 \end{bmatrix}$$

Следовательно, результаты сравнения двух итоговых результирующих матриц на примерах 13 и 14, полученных в результате работы процедур *getBetterResultMatrix* и *modifyRR*, разные. Значит, эвристические правила (см. п. 4.3) не всегда срабатывают для получения лучшей матрицы среди итоговых результирующих матриц. Для этого протестируем разработанную версию RR *modifyRR* и выясним, насколько (в процентном выражении) получаются более хорошие результаты.

5.1 Оценка эвристических правил

Для тестирования процедуры *modifyRR* мы реализовали процедуру генерации операторных матриц *GenerateMatrixRR* с выбором неоднозначности, которые возникнут во время работы алгоритма RR. Генератор был реализован с помощью обратного хода алгоритма RR, т. е. по исходной сгенерированной или по предложенной матрице применяем шаги алгоритма RR в обратном порядке, чтобы получить матрицу с желаемыми параметрами.

Процедура *GenerateMatrixRR* принимает девять входных параметров:

- m — размер матрицы $m \times m$;
- r — максимальный порядок строк матрицы;
- *Iter* — количество итераций работы алгоритма RR;
- *isAmbiguity* — наличие неоднозначности;
- *numberAmbiguity* — выбор неоднозначности, если *isAmbiguity* это *true*;
- *rangeOrderPolynomials* — максимальная степень полиномов;
- $rangeNumbers$ — коэффициенты полинома из диапозона $[-rangeNumbers \dots rangeNumbers]$;
- *bOptionalMatrix* — *true*, если обход алгоритма RR в обратном порядке начнётся с введённой матрицы, у которой порядок строк дифференцирования сгенерированной матрицы окажется намного больше чем r, и сами характеристики коэффициента при порядке дифференцирования будут громоздкими. Иначе *false*, чтобы

программа сама сгенерировала матрицу с желаемыми параметрами, у которой порядок дифференцирования строк — будет равен r;

- *optionalMatrix* — введённая добавочная матрица.

В результате процедура вернёт операторную матрицу.

Для каждых параметров m, r, $Iter$ при $m = \{3, 4, 6, 8\}$, $r = \{3, 6, 8, 10\}$, $Iter = \{1, 2, 4\}$ и $bOptionalMatrix = false$, и для некоторых комбинаций параметров m, r, $Iter$ при $bOptionalMatrix = true$ было сгенерировано по 20 операторных матриц. Порядки дифференцирования и их коэффициенты были сгенерированы случайным образом. Коэффициенты при порядках дифференцирования — случайные полиномы, степени которых не больше 20. Коэффициенты полиномов выбирались из диапазона $[-50 \ldots 50]$. Для всего этого использовались процедуры *Generate* из пакета *RandomTools*, *RandomMatrix* из пакета *LinearAlgebra* и стандартая процедура *randpoly*. Матрицы генерировались разрежёнными, каждый элемент был ненулевым с вероятностью $\frac{1}{[m]+1}$, где $[\ldots]$ — целая часть от числа. Во время тестирования фиксировались параметры:

1. *isEqualMat* — процент совпадения матриц, полученных в результате работы процедур *modifyRR* и *getBetterResultMatrix*.

2. *CountIter* — процент достижения итоговых результирующих матриц за разное количество итераций.

3. *CountM* — среднее количество итоговых результирующих матриц на один тест, полученных во время работы алгоритма RR.

4. $MaxDiff$ — порядок дифференцирования, полученный в результате генерации матрицы с параметром $bOptionalMatrix = true$.

Результаты экспериментов представлены в таблицах 4 и 5, содержащих информацию о параметрах. Таблицы были поделены по значениям *bOptionalMatrix*, т. к. при $bOptionalMatrix = true$ количество неоднозначностей, возникающих во время работы алгоритма становится больше. В этом можно убедится по показателю *CountM* таблицы 5, т. е. *CountM* равен среднему числу неоднозначностей, возникших в ходе работы алгоритма. Результаты показывают, что на более чем 900 тестов из таблицы 4 и на 200 тестах из таблицы 5 проценты совпадения матриц с наиболее хорошими характеристиками равны 0.74 и 0.57 соответственно. Можно смело утверждать, что эвристические правила процедуры *modifyRR* в большинстве случаев применимы на практике, если элементы

операторных матриц не состоят из громоздких полиномов, т. е. из большого количества многочленов, большого количества степеней многочленов и чисел, стоящих при многочленах, в коэффициентах при порядке дифференцирования. Это справедливо, потому что во время проведения экспериментов при $bOptionalMatrix = false$ сгенерированные операторные матрицы не имели больших характеристик по значению, в отличие от матриц, полученных от генератора операторных матриц при $bOptionalMatrix = true$.

Кроме того, приведённые результаты показывают, что с увеличением основных показателей m и r из таблицы 5 растёт *CountIter*.

Вычислим процент совпадения матриц, полученных в результате работы процедур *modifyRR* и *getBetterResultMatrix*, по всем тестам. Количество строк с различными параметрами из таблиц 4 и 5 равно 36 и 10 соответственно. Суммы по *isEqualMat* — равны 26.75 и 5.7. Значит процент совпадения матриц по всем тестам равен

$$\frac{26.75 + 5.7}{36 + 10} = 0.705.$$

Таблица 4. Статистика при $bOptionalMatrix = false$

m	r	*Iter*	*isEqualMat*	*CountIter*	*CountM*
3	3	1	0.85	0	4.75
3	3	2	0.85	0	8.5
3	3	4	0.85	0	4.75
3	6	1	0.85	0	6.5
3	6	2	0.85	0	6.5
3	6	4	0.75	0.05	6.5
3	10	1	0.75	0.1	5.55
3	10	2	0.75	0.05	5.95
3	10	4	0.75	0.05	5.95
4	3	1	0.7	0	3.8
4	3	2	0.6	0	5.6
4	3	4	0.7	0	4.8
4	6	1	0.75	0.1	9.6
4	6	2	0.75	0	7.85
4	6	4	0.75	0	10.25
4	10	1	0.75	0	7.35

Таблица 4. Статистика при $bOptionalMatrix = false$

m	r	$Iter$	$isEqualMat$	$CountIter$	$CountM$
4	10	2	0.7	0	9.15
4	10	4	0.75	0	11
6	3	1	0.7	0	8.1
6	3	2	0.6	0	8.2
6	3	4	0.7	0	15.25
6	6	1	0.75	0.05	6.6
6	6	2	0.85	0	8.95
6	6	4	0.95	0	3.7
6	10	1	0.75	0	7.2
6	10	2	0.65	0	8.1
6	10	4	0.6	0.05	19.5
8	3	1	0.8	0	12.6
8	3	2	0.8	0	37.6
8	3	4	0.7	0	38.3
8	6	1	0.4	0.05	40.45
8	6	2	0.85	0	62.5
8	6	4	0.8	0	19.7
8	10	1	0.75	0	11.9
8	10	2	0.75	0	21.35
8	10	4	0.65	0	23.35
			0.74	0.0138	

Таблица 5. Статистика при $bOptionalMatrix = true$

m	r	$Iter$	$isEqualMat$	$CountIter$	$CountM$	$MaxDiff$
3	3	1	0.85	0	4.75	6
3	3	2	0.7	0.1	17.5	8
3	6	1	0.65	0.15	14.55	10
3	6	2	0.45	0.4	197.2	9
4	3	1	0.7	0.1	11.5	5
4	3	2	0.45	0.3	683.7	6

Таблица 5. Статистика при $bOptionalMatrix = true$

m	r	$Iter$	$isEqualMat$	$CountIter$	$CountM$	$MaxDiff$
4	6	1	0.6	0.35	125.15	10
6	3	1	0.55	0.15	37.75	6
6	3	2	0.35	0.4	491.25	8
8	3	1	0.4	0.45	682.3	13
			0.57	0.24		

Заключение

В ходе работы был проанализирован алгоритм Row-Reduction и выявлены неоднозначности выбора, возникающие в ходе его работы. В системе компьютерной алгебры Maple с использованием пакетов OreTools, LinearAlgebra и RandomTools была реализована программа *outputRR*, собирающая в единый список все возможные результаты работы алгоритма RR по заданной операторной матрице. Проанализировав все результирующие матрицы, были выработаны критерии сравнения для определения более лучшего (простого) вида операторных матриц. Также были сформулированы эвристические правила разрешения неоднозначного выбора в ходе работы алгоритма для получения результатов более простого вида. Во время проведения экспериментов было выявлено, что реализация алгоритма RR *modifyRR*, использующая эвристические правила, выполняется за меньшее количество итераций.

Для оценки предложенных эвристик была проведена серия экспериментов. Для этого использовался разработанный генератор операторных матриц, который по заданным параметрам выдаёт операторную матрицу с желаемыми характеристиками. В ходе экспериментов результат работы *modifyRR* сравнивался с результатом работы процедуры *getBetterResultMatrix*, которая всегда выбирает лучшую матрицу из всех возможных итоговых результирующих матриц. Было проведено более тысячи сравнительных тестов, в ходе которых было определено, что в 70.5% случаях процедура *modifyRR* возвращает лучший из возможных результатов.

Список литературы

[1] В. С. Муха Вычислительные методы и компьютерная алгебра. // 2-е изд., испр. и доп.. — Минск: БГУИР, 2010.

[2] *J. von zur Gathen, J. Gerhard* Modern Computer Algebra, Third Edition. // Cambridge University Press. — 2013. — 808 p. — ISBN 978-1-107-03903-2.

[3] В. М. Зюков Компьютерная алгебра // Издательство Томского университета, 2014.

[4] *Таранчук В. Б.* Основные функции систем компьютерной алгебры. // Минск: БГУ, 2013.

[5] *Е. А. Ильченко* Инструменты математического сервиса MathPartner для выполнения параллельных вычислений на кластере // Труды ИСП РАН, том 28, вып. 3, 2016 г.. — с. 173-188. DOI: 10.15514/ISPRAS-2016-28(3)-11.

[6] *С. А. Абрамов, А. А. Рябенко, Д. Е. Хмельнов* Выявляющие матрицы линейных дифференциальных систем произвольного порядка // Программирование, №2, 2017. — с. 7-16.

[7] *S.A. Abramov, M.A. Barkatou* On the Dimension of Solution Spaces of Full Rank Linear Differential Systems. // CASC 2013, LNCS 8136, 2013. — Pp. 1–9.

[8] *С. А. Абрамов* Элементы компьютеной алгебры линейных дифференциальных, разностных и q-разностный операторов. // Изд.-во МНЦМО, 2012.

[9] С. А. Абрамов, Х. К. Ле (H. Q. Le), З. Ли (Z. Li) *Кольца полиномов Оре одной переменной в компьютерной алгебре.* // Современная математика и приложения, т.13, 2004. — с.24-39.

[10] *Beckermann M., Cheng H., Labahn G.* Fraction-free row reduction of matrices of ore polynomials // J. of Symbolic Computation. — 2006. — vol. 41 (5). — Pp. 513–543.

[11] *Barkatou M., El Bacha C., Labahn G., Pflügel E.* On simultaneous row and column reduction of higher-order linear differencial systems // J. of Symbolic Computation. — 2013. — vol. 49. — Pp. 45–64.

[12] *Панферов А.А.* Системы дифференциальных уравнений с выделенной частью неизвестных. // Программирование, № 2, 2015. — с. 26–36.

Приложение А

Пример вычисления результирующих матриц

Пример вычисления результирующей операторной матрицы, где в процессе её нахождения алгоритмом Row-Reduction проявляются обе выделенные неоднозначности: неоднозначный выбор среди нескольких векторов коэффициентов линейной зависимости строк фронтальной матрицы и неоднозначный выбор строки операторной матрицы для последующей редукции (шаг 2 алгоритма).

$$
\left[\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,2-x,5) & \mathit{OrePoly}(0,-x,10x) & \mathit{OrePoly}\left(0,-x^2-x\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],\right.
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,-3,-6) & \mathit{OrePoly}(3x,1) & \mathit{OrePoly}(-3x,1) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left.\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,-3) & \mathit{OrePoly}(2+3x,1) & \mathit{OrePoly}(2-3x,1) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right]\right],
$$

$$
\left[\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,2-x,5) & \mathit{OrePoly}(0,-x,10x) & \mathit{OrePoly}\left(0,-x^2-x\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],\right.
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,-3,-6) & \mathit{OrePoly}(3x,1) & \mathit{OrePoly}(-3x,1) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left.\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,3x-6) & \mathit{OrePoly}(0,5+3x,-30x) & \mathit{OrePoly}\left(0,3x^2+3x+5\right) \\ \mathit{OrePoly}(0,-3,-6) & \mathit{OrePoly}(3x,1) & \mathit{OrePoly}(-3x,1) \\ \mathit{OrePoly}(0,-3) & \mathit{OrePoly}(2+3x,1) & \mathit{OrePoly}(2-3x,1) \end{array}\right]\right],
$$

$$
\left[\left[\begin{array}{ccc} \mathit{OrePoly}(0,0,2-x,5) & \mathit{OrePoly}(0,-x,10x) & \mathit{OrePoly}\left(0,-x^2-x\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],\right.
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,5,8+x) & \mathit{OrePoly}(-5x,x,-10x) & \mathit{OrePoly}\left(5x,x^2+x\right) \\ \mathit{OrePoly}(0,1,2,1) & \mathit{OrePoly}(-x) & \mathit{OrePoly}(x) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left[\begin{array}{ccc} \mathit{OrePoly}(0,5,8+x) & \mathit{OrePoly}(-5x,x,-10x) & \mathit{OrePoly}\left(5x,x^2+x\right) \\ \mathit{OrePoly}(0,-3,-6) & \mathit{OrePoly}(3x,1) & \mathit{OrePoly}(-3x,1) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right],
$$

$$
\left.\left[\begin{array}{ccc} \mathit{OrePoly}(0,5,8+x) & \mathit{OrePoly}(-5x,x,-10x) & \mathit{OrePoly}\left(5x,x^2+x\right) \\ \mathit{OrePoly}(0,-3) & \mathit{OrePoly}(2+3x,1) & \mathit{OrePoly}(2-3x,1) \\ \mathit{OrePoly}(0,0,3) & \mathit{OrePoly}(1) & \mathit{OrePoly}(1) \end{array}\right]\right],
$$

$$\left[\left[\begin{bmatrix} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly(0,-x^2-x) \\ OrePoly(0,1,2,1) & OrePoly(-x) & OrePoly(x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},\right.\right.$$
$$\begin{bmatrix} OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,1,2,1) & OrePoly(-x) & OrePoly(x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\begin{bmatrix} OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,-3,-6) & OrePoly(3x,1) & OrePoly(-3x,1) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\left.\begin{bmatrix} OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,-3,-6) & OrePoly(3x,1) & OrePoly(-3x,1) \\ OrePoly(0,-3) & OrePoly(2+3x,1) & OrePoly(2-3x,1) \end{bmatrix}\right],$$
$$\left[\begin{bmatrix} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly(0,-x^2-x) \\ OrePoly(0,1,2,1) & OrePoly(-x) & OrePoly(x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},\right.$$
$$\begin{bmatrix} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly(0,-x^2-x) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\begin{bmatrix} OrePoly(0,0,3x-6) & OrePoly(0,5+3x,-30x) & OrePoly(0,3x^2+3x+5) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\left.\begin{bmatrix} OrePoly(0,-15) & OrePoly(10+15x,5) & OrePoly(10-15x,5) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix}\right],$$
$$\left[\begin{bmatrix} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly(0,-x^2-x) \\ OrePoly(0,1,2,1) & OrePoly(-x) & OrePoly(x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},\right.$$
$$\begin{bmatrix} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly(0,-x^2-x) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\begin{bmatrix} OrePoly(0,0,3x-6) & OrePoly(0,5+3x,-30x) & OrePoly(0,3x^2+3x+5) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly(5x,x^2+x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix},$$
$$\left.\left.\begin{bmatrix} OrePoly(0,0,3x-6) & OrePoly(0,5+3x,-30x) & OrePoly(0,3x^2+3x+5) \\ OrePoly(0,-15) & OrePoly(10+15x,5) & OrePoly(10-15x,5) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{bmatrix}\right]\right],$$

$$\left[\left[\begin{array}{ccc} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly\left(0,-x^2-x\right) \\ OrePoly(0,1,2,1) & OrePoly(-x) & OrePoly(x) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{array}\right],\right.$$

$$\left[\begin{array}{ccc} OrePoly(0,0,2-x,5) & OrePoly(0,-x,10x) & OrePoly\left(0,-x^2-x\right) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly\left(5x,x^2+x\right) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{array}\right],$$

$$\left[\begin{array}{ccc} OrePoly(0,0,3x-6) & OrePoly(0,5+3x,-30x) & OrePoly\left(0,3x^2+3x+5\right) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly\left(5x,x^2+x\right) \\ OrePoly(0,0,3) & OrePoly(1) & OrePoly(1) \end{array}\right],$$

$$\left.\left[\begin{array}{ccc} OrePoly(0,0,3x-6) & OrePoly(0,5+3x,-30x) & OrePoly\left(0,3x^2+3x+5\right) \\ OrePoly(0,5,8+x) & OrePoly(-5x,x,-10x) & OrePoly\left(5x,x^2+x\right) \\ OrePoly(0,-15) & OrePoly(10+15x,5) & OrePoly(10-15x,5) \end{array}\right]\right]$$