Московский государственный университет имени М. В. Ломоносова
Казахстанский филиал
Кафедра алгоритмических языков

Колесников Владислав Владимирович

Реализация обобщенного алгоритма Абрамова-Бронштейна в системе компьютерной алгебры Мэйпл

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
Панфёров А. А.

Москва, 2018

Содержание

Введение

Рассматриваются линейные однородные дифференциальные системы. Более формально рассматриваемый тип систем будет описан в разделе 2 данной работы. Зафиксируем одну из таких систем и обозначим через y вектор неизвестных функций этой системы.

При решении некоторых задач отдельные компоненты вектора неизвестных y могут представлять больший интерес. Такие компоненты будем называть *выделенными*. К задачам, в которых некоторые неизвестные представляют больший интерес, например, относятся: проверка существования решений, выделенные компоненты которых принадлежат заданным классам; поиск решений только для выделенных компонент; частичная устойчивость по выделенным компонентам и другие. Будем считать, что в рассматриваемой системе часть неизвестных были нами отмечены как выделенные.

Будем считать, что рассматриваемая система является системой первого порядка и запишем ее в виде

$$A_1 y' + A_0 y = 0 \tag{1}$$

В разделе 2.1 будет показано, что всякую линейную однородную дифференциальную систему некоторого фиксированного порядка r, записанную в виде

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

всегда можно свести к линейной однородной дифференциальной системе первого порядка вида (1), поэтому рассмотрение системы первого порядка не ограничивает общности рассуждений относительно систем произвольных порядков.

В случае, если матрица A_1 рассматриваемой системы обратима, то можно перейти к нормальной системе

$$y' = Ay, \tag{3}$$

где $A = -A_1^{-1} A_0$.

С. А. Абрамовым и М. Бронштейном для нормальных систем вида (3) в работе [1] был предложен алгоритм, именуемый AB-алгоритмом, позволяющий для выделенных компонент вектора неизвестных y по исходной системе построить новую нормальную систему, в которую входят только выделенные компоненты y и, возможно, некоторые их производные. Новая система, как показано в [1], обладает свойствами, позволяющими решать упомянутые ранее задачи.

AB-алгоритм был реализован в `Maple` в виде процедуры `ReducedSystem` и его реализация присутствует в стандартной поставке `Maple`.

В случае же вырожденности матрицы A_1 системы (1) перейти к системе вида (3) нельзя, что делает невозможным прямое применения AB-алгоритма к таким системам. Панфёровым А. А. был разработан и приведен в работе [2] алгоритм `Extract`, позволяющий по системам первого порядка вида (1) с необратимой матрицей A_1 построить новую нормальную систему, которая, затем, может быть передана AB-алгоритму для последующей обработки.

Алгоритм `Extract` так же был реализован в `Maple` и его реализация может быть найдена по адресу `www.ccas.ru/ca/extract`.

В работе [3] был предложен, но до сих пор не реализован, алгоритм `ExtrAB`, который объединяет в себе алгоритм `Extract` и AB-алгоритм. Этот алгоритм принимает на вход произвольную систему первого порядка вида (1) и результатом его применения является нормальная система, в которую не входят невыделенные неизвестные.

Таким образом, с учетом возможности сведения линейной однородной системы произвольного порядка к системе первого порядка, можно говорить, что алгоритм `ExtrAB` обобщает AB-алгоритм на случай произвольных линейных однородных дифференциальных систем (на самом деле, на системы, подающиеся на вход алгоритмам `Extract` и `ExtrAB` накладывается одно дополнительное требование, которое будет уточнено в разделе 2.1).

Реализации универсального алгоритма `ExtrAB` и посвящена данная выпускная квалификационная работа.

1 Постановка задачи

Было принято решение осуществлять реализацию алгоритма `ExtrAB` на основе существующих реализаций алгоритма `Extract` и AB-алгоритма. Процедурам, реализующим эти алгоритмы, исходные дифференциальные системы подаются на вход в виде матриц коэффициентов. Результирующие нормальные системы они выдают так же в виде матриц коэффициентов этих системы. Однако, более традиционным способом задания дифференциальных систем в `Maple` является набор (список или множество) явно заданных дифференциальных уравнений.

Операция дифференцирования в `Maple` задается несколькими способами. Самым распространенным (и принятым в данной работе) из них является команда

$$\texttt{diff(f(x),x}_1\texttt{, ..., x}_j\texttt{).}$$

Она используется для поиска частных производных функции $f(x)$ по $x_1, \ldots, x_j$. Наиболее часто эта команда используется в виде `diff(f(x),x)`, что означает производную функции $f(x)$ по x. Например:

```
> diff(sin(x),x)
```

$$cos(x)$$

Если `diff` передается неизвестная функция, то вызов команды `diff` возвращает сам себя без вычисления:

```
> diff(f(x),x)
```

$$\frac{df(x)}{dx}$$

Таким образом, с помощью команды `diff` дифференциальную систему можно задать, например, так:

```
> odesys := [diff(y(x),x) = 0, diff(y(x),x$2)+2diff(y(x),x) = 0]
```

$$\texttt{odesys := } [\tfrac{dy(x)}{dx} = 0,\ \tfrac{d^2y(x)}{dx^2} + 2\tfrac{dy(x)}{dx} = 0]$$

Большинство процедур `Maple` по работе с дифференциальными системами ожидают на вход и выдают в качестве результата системы именно в таком представлении, поэтому будем называть такое представление стандартным для `Maple`, а системы заданные в таком виде системами, заданными в стандартном для `Maple` виде.

В связи с тем, что многие процедуры обработки дифференциальных систем в `Maple` принимают на вход и выдают в качестве результата системы, заданные в стандартном для `Maple` виде, для удобства использования реализации алгоритма `ExtrAB` с этими процедурами была поставлена задача реализовать алгоритм `ExtrAB` таким образом, чтобы системы, принимаемые на вход и возвращаемые в качестве результата, были представлены в стандартном для `Maple` виде. А так как было принято решение реализацию осуществлять на основе имеющихся реализаций алгоритма `Extract` и AB-алгоритма, принимающих на вход и выдающих в качестве результата системы в матричном виде, встала необходимость в реализации вспомогательной процедуры преобразования исходной системы из стандартного для `Maple` вида в матричный и обратно.

Так же, был принят во внимание тот факт, что процедуры, реализующие алгоритм `Extract` и AB-алгоритм, приспособлены для работы с системами первого порядка. И хотя преобразование системы произвольного порядка в систему первого порядка является довольно простым процессом, было бы удобно в реализации `ExtrAB` обрабатывать этот случай.

Для обработки случая дифференциальных систем произвольного порядка, таким образом, встала необходимость в реализации вспомогательной процедуры сведения исходной системы произвольного порядка, заданной в матричном виде, к системе первого порядка, заданной так же в матричном виде.

Целью выпускной квалификационной работы является реализация в системе компьютерной алгебры `Maple` алгоритма `ExtrAB`, обобщающего AB-алгоритм на случай произвольных линейных однородных дифференциальных систем. Для этого:

- реализовать процедуру преобразования дифференциальной системы, заданной в стандартном для `Maple` виде в матричный вид;
- реализовать процедуру сведения системы произвольного порядка, заданной в матричном виде, к системе первого порядка;
- реализовать на основе существующих реализаций алгоритма `Extract` и AB-алгоритма, а также вспомогательных процедур, основную процедуру `ExtrAB`, принимающую на вход линейную однородную дифференциальную систему произвольного порядка, заданную в стандартном для `Maple` виде, и возвращающую новую нормальную систему, в которую не входят невыделенные неизвестные и которая задана в стандартном для `Maple` виде.

2 Теоретическая часть

2.1 Предварительные сведения

Пусть K — дифференциальное поле характеристики 0 с производной $\partial = '$. Обозначим через $K[\partial]$ кольцо дифференциальных операторов с коэффициентами из K. Рассматриваются линейные однородные системы обыкновенных дифференциальных уравнений, представимые в виде

$$L(y) = 0, \tag{4}$$

где $L \in K[\partial]^{m\times m}$, а $y = (y_1, \dots, y_m)^T$ — вектор неизвестных функций.

Например, нормальная система вида (3) представима в виде (4) при $L = I\partial - A$.

Будем считать некоторую часть компонент вектора неизвестных $\{y_{i_1}, \dots, y_{i_k}\} \subset \{y_1, \dots, y_m\} (1 \le k \le m)$ выделенными.

Линейная однородная дифференциальная система произвольного порядка вида (2) может быть записана в операторном виде (4), при

$$L = A_r\partial^r + A_{r-1}\partial^{r-1} + \cdots + A_1\partial + A_0,$$

где $r \in \mathbb{N}, A_i \in K^{m\times m}$. Число r называется *порядком* операторной матрицы L (системы (2)), а матрица A_r называется *ведущей матрицей* L (системы (2)).

Пусть $L \in K[\partial]^{m\times m}$. Через $L_{i,*}$ обозначим i-ю строку матрицы L. Пусть $J \subseteq \{1, \dots, m\}$, тогда строки $L_{i,*}$, где $i \in J$, называются $K[\partial]^{m\times m}$-*линейно зависимыми*, если существуют дифференциальные операторы $\{W_i\} \subset K[\partial]^{m\times m}$, не все одновременно равные нулю, такие, что $\sum_{i\in J} W_i L_{i,*} = 0$; в противном случае они называются $K[\partial]^{m\times m}$-*линейно-независимыми.*

Ранг операторной матрицы L определяется как максимальное число ее $K[\partial]^{m\times m}$-линейно-независимых строк. Дифференциальная система вида (4) имеет *полный ранг* тогда и только тогда, когда ранг ее операторной матрицы L равен m.

Требование полноранговости и есть то самое требование к дифференциальным системам, подающимся на вход алгоритмам `Extract` и `ExtrAB`, упоминавшееся во введении. С учетом этого, более точно будет сказать, что алгоритм `ExtrAB` обобщает AB-алгоритм на случай произвольных линейных однородных дифференциальных систем полного ранга.

Будем считать, что все рассматриваемые далее линейные однородные системы имеют полный ранг.

Покажем теперь, что системы прозвольного порядка вида (2) всегда могут быть сведены к системам первого порядка вида (1). В самом деле, переходя к вектору неизвестных $Y = (y_1, \dots, y_m, y_1', \dots, y_m', \dots, y_1^{(r-1)}, \dots, y_m^{(r-1)})^T$ и полагая

$$\tilde{A}_1 = \begin{bmatrix} I_m & & & 0 \\ & \ddots & & \\ & & I_m & \\ 0 & & & A_r \end{bmatrix}, \ \tilde{A}_0 = \begin{bmatrix} 0 & -I_m & & 0 \\ & \ddots & \ddots & \\ 0 & & 0 & -I_m \\ A_0 & A_1 & \dots & A_{r-1} \end{bmatrix},$$

где I_m — единичная матрица $m \times m$, исходная система сводится к системе первого порядка

$$\tilde{A}_1 Y' + \tilde{A}_0 Y = 0.$$

Учитывая данное замечание, будем далее, не ограничивая общности, считать, что имеем дело только с системами первого порядка вида (1).

2.2 AB-алгоритм

Как уже отмечалось во введении, если ведущая матрица A_1 системы (1) обратима, то можно перейти к нормальной системе вида (3). Применяя к полученной нормальной системе AB-алгоритм на выходе получается новая нормальная система

$$z' = Rz, \tag{5}$$

где компонентами z являются только выделенные компоненты y и, возможно, некоторые их производные.

Пример 1. Рассмотрим следующую систему:

$$y' = \begin{bmatrix} -1 & 1 & -2 \\ 4 & 1 & 0 \\ 5/2 & 1 & -1 \end{bmatrix} y,$$

где $y = (y_1, y_2, y_3)^T$. Пусть y_1 — выделена. В результате применения к этой системе AB-алгоритма получится система

$$z' = \begin{bmatrix} 0 & 1 \\ -2 & -2 \end{bmatrix} z,$$

где $z = (y_1, y_1')^T$.

Процедура `ReducedSystem`, реализующая AB-алгоритм, является настраиваемой относительно типа исходной системы. Это достигается путем привлечения некоммутативных полиномов Оре (об их использовании в компьютерной алгебре см., например, [4], [5]). Поэтому перед использованием `ReducedSystem` необходимо выполнить подготовительную работу, а именно — зафиксировать кольцо полиномов Оре. Например для дифференциального случая это делается так:

```
> R := OreTools:-SetOreRing(x, 'differential'):
```

Процедура `ReducedSystem` имеет следующий прототип

```
ReducedSystem(M, ns, R),
```

где `M` — матрица, задающая исходную нормальную систему (3), `ns` — множество индексов выделенных неизвестных, `R` — кольцо полиномов Оре.

Тогда пример 1 в `Maple` будет выглядеть следующим образом (будем считать, что `R` — задано):

Пример 1*.

```
> M := Matrix([[-1, 1, -2], [4, 1, 0], [5/2, 1, -1]]):
> OreTools:-Consequences:-ReducedSystem(M, 1, R)
```

$$\left[\begin{bmatrix} 0 & 1 \\ -2 & -2 \end{bmatrix}, \ \{[1, 1]\} \right]$$

Результат работы процедуры представляет из себя список из двух элементов. Первый элемент это матрица полученной нормальной системы, второй элемент это множество соответствий индексов выделенных неизвестных в исходной системе и в результирующей.

2.3 Алгоритм `Extract`

Однако, как отмечалось во введении, если ведущая матрица системы первого порядка (1) необратима, то к этой системе AB-алгоритм напрямую применить нельзя. Такие системы называются *дифференциально-алгебраическими*.

Применение к системе такого вида алгоритма `Extract`, описанного в [2], позволяет получить новую нормальную систему

$$\tilde{y}\,' = B\tilde{y}, \tag{6}$$

где $\tilde{y} \subset y$. В вектор неизвестных $\tilde{y}$ полученной системы (6) входит часть выделенных и часть невыделенных неизвестных. Алгоритм так же на выходе дает информацию о выражении невошедних в результирующую систему выделенных компонент через вошедшие.

Пример 2. Рассмотрим систему

$$\begin{bmatrix} 1 & 0 & 0 & x^2 \\ 0 & 2x & 0 & 0 \\ 0 & 0 & x^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} y' + \begin{bmatrix} 1 & x & 0 & 2x \\ 0 & 2 & 0 & 0 \\ 0 & 1 & -x & 0 \\ 1 & 0 & 0 & x^2 \end{bmatrix} y = 0$$

с вектором неизвестных $y = (y_1, y_2, y_3, y_4)^T$, выделенными будем считать y_1, y_2. Результатом применения `Extract` к этой системе являются новая дифференциальная система

$$\tilde{y}' = \begin{bmatrix} -1/x & 0 \\ -1/x^2 & 1/x \end{bmatrix} \tilde{y},$$

где $\tilde{y} = (y_2, y_3)^T$, а также алгебраическая система из одного уравнения

$$y_1 = -x \cdot y_2,$$

задающая выражение невошедшей в новую дифференциальную систему выделенной неизвестной y_1 через вошедную неизвестную y_2.

На вход алгоритму `Extract` подается дифференциально-алгебраическая система первого порядка вида (1), соответствующая операторная матрица которой является приведённой по строкам. Понятие приведенности операторной матрицы по строкам дается в работе [2] и более подробно в [6] и в данной работе не так существенно. Также на вход алгоритма подается множество выделенных неизвестных

В качестве результата алгоритм возвращает матрицы новой дифференциальной и алгебраической систем.

Схематически алгоритм `Extract` может быть изображен так:

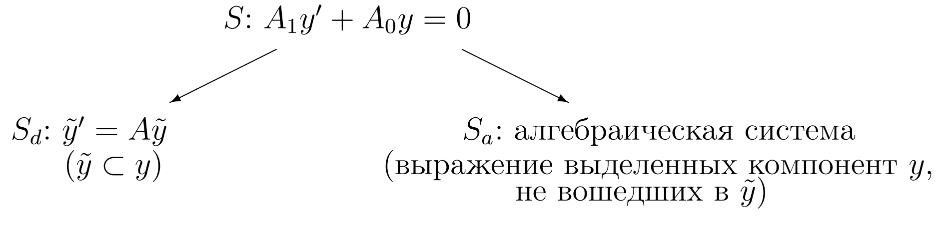

Рис. 1: Extract

Для работы процедуры `Extract` производится та же подготовительная работа по заданию кольца полинов Оре. Прототип процедуры `Extract` выглядит следующим образом:

```
Extract(A1, A0, ns, R),
```

где `A1, A0` — матрицы исходной дифференциально-алгебраической системы, `ns` — множество индексов выделенных неизвестных, `R` — кольцо полиномов Оре.

Пример 2 может быть переписан в `Maple` следующим образом:

Пример 2*.

```
> A1 := Matrix([[1, 0, 0, x^2], [0, 2x, 0, 0], [0, 0, x^2, 0], [0, 0, 0, 0]]):
> A0 := Matrix([[1, x, 0, 2x], [0, 2, 0, 0], [0, 1, -x, 0], [1, 0, 0, x^2]]):
> Extract(A1,A0,{1,2},R)
```

$$\left[\begin{bmatrix} -1/x & 0 \\ -1/x^2 & 1/x \end{bmatrix},\ \ \{[2,1]\},\ \ \begin{bmatrix} -x \end{bmatrix},\ \ \{[1,1]\}\right]$$

Результат состоит из четырех элементов. Первый элемент это матрица результирующей нормальной системы, второй элемент это множество пар соответствий индексов выделенных неизвестных в исходной системе и в полученной, третий и четвертый элементы задают алгебраическую систему, по которой получается выражение невошедшей в новую систему неизвестной через вошедшую.

2.4 Алгоритм `ExtrAB`

В примере 2 видно, что в полученную в результате работы алгоритма `Extract` нормальную систему (6) могут входить невыделенные неизвестные. Чтобы это исправить и иметь возможность решать упомянутые во введении задачи, можно к ней далее применить AB-алгоритм и таким образом получить систему, в которую уже не будут входить лишние неизвестные. Такое последовательное применение указанных алгоритмов было описано в [3] и было названо алгоритмом `ExtrAB`.

Схематически применение алгоритма `Extract` с последующим применением AB-алгоритма выглядит следующим образом:

$$\begin{tikzcd}
 & S\colon A_1y' + A_0y = 0 \arrow[dl] \arrow[dr] & \\
\begin{array}{c} S_d\colon \tilde{y}' = A\tilde{y} \\ (\tilde{y} \subset y) \end{array} \arrow[d, "\text{AB-алгоритм}"] & & \begin{array}{c} S_a\colon \text{алгебраическая система} \\ (\text{выражение выделенных компонент } y, \\ \text{не вошедших в } \tilde{y}) \end{array} \\
z' = Bz & &
\end{tikzcd}$$

Рис. 2: Extract + AB

Назовем наше множество выделенных неизвестных s. В [3] было дано следующее определение: системы S_d, S_a называются *согласованными* с (S, s), если проекция пространства решений S на s совпадает с проекцией на выделенные неизвестные пространства решений систем S_d, S_a в произвольном расширении исходного дифференциального поля.

Общий алгоритм `ExtrAB` может быть схематически изображен так:

$$S\colon A_1 y' + A_0 y = 0$$

$S_d^{AB}\colon z' = Bz$
(z — часть выделенных компонент y
и некоторые их производные)

S_a: алгебраическая система
(выражение выделенных компонент y,
не вошедших в $\tilde{y}$)

Рис. 3: ExtrAB

В [3] была дана теорема, в которой утверждается, что получаемые алгоритмом системы S_d^{AB} и S_a являются согласованными с (S, s), что позволяет решать упомянутые во введении задачи.

Рисунки 1, 2 и 3 были заимствованы из [3].

3 Практическая часть

3.1 Процедура `ODESYSconvert`

Для преобразования линейной однородный системы обыкновенных дифференциальных уравнений из стандартного для `Maple` вида в матричный вид (2) потребовалась реализация вспомогательной процедуры `ODESYSconvert`.

На вход ей подается исходная система `eqns` в виде множества или списка обыкновенных дифференциальных уравнений. Предполагается, что в системе присутствует только одна независимая переменная. Внутри программы после выполнения необходимых подготовительных работ в первую очеред с помощью стандартной процедуры `indets` производится выявление единственной независимой переменной системы, обозначим ее через x. Далее с помощью этой же процедуры и уже выявленной независимой переменной x строится вектор неизвестных функций системы, зависящих от этой независимой переменной. Обозначим через m число неизвестных функций, а сам вектор неизвестных обозначим $y = (y_1, \dots, y_m)^T$.

Затем процедура находит порядок r входной системы `eqns` с помощью процедуры `ODESYSorder(eqns)`, реализованной самостоятельно ввиду отсутствия в существующих пакетах `Maple` аналога, и затем в цикле по i от r до 0 производит следующие шаги:

1. оставляет в левой части уравнения только члены, соответствующие производным порядка i функций вектора неизвестных, а остальные слагаемые, соответствующие производным порядка, меньшего i, переносит в правые части уравнений (соответственно, их знаки меняются на противоположные);

2. приводит подобные в получившихся левых частях и строит матрицу A^i размера $m \times m$, где A^i_{kp} соответствует коэффициенту при $y_p^{(i)}(x)$ в k-м уравнении;

3. добавляет полученную на шаге 2 матрицу A^i в начало результирующего списка A;

4. отбрасывает левые части уравнений; знаки слагаемых в правых частях меняются на противоположные и они считаются новыми левыми частями уравнений.

Шаги 1,2 производятся автоматически с помощью процедуры `GenerateMatrix` из пакета `LinearAlgebra`.

Сформированный в процессе работы программы список $A = [A_0, A_1, \dots, A_r]$ матричных коэффициентов задает исходную систему в матричном виде (2) и подается процедурой на выход.

Встроенная роцедура `indets(expr, type)` возвращает список имен всех неизвестных, вошедших в `expr`. Если задано значение `type`, то выдается список только неизвестных, принадлежащих этому типу. Значениями `type`, например, могут быть `function` или `constant`.

Например:

> indets(5x$-$3sin(y)+xy^4+e$^{z^2}$,function)

$$\{e^{z^2}, \sin(y)\}$$

Вспомогательная процедура `ODESYSorder` находит порядок переданной ей системы дифференциальных уравнений, заданной в стандартном для `Maple` виде, как максимум из порядков входящих в нее дифференциальных уравнений. Порядок каждого уравнения находится с помощью процедуры `difforder(a)` из пакета `PDEtools`.

Процедура ODESYSorder имеет следующий прототип:

```
ODESYSorder(eqns),
```

где eqns — список или множество дифференциальных уравнений.

Приме вызова процедуры ODESYSorder:

> eq2 := $\frac{d^4}{dx^4}g(x) = 0$

$$eq2 := \frac{d^4}{dx^4}g(x) = 0$$

> ODESYSorder([eq1,eq2])

$$4$$

Рассмотрим процедуру GenerateMatrix(eqns,vars). Ее первый параметр eqns это список или множество уравнений или выражений, ее второй параметр vars — переменные этих уравнений. Каждое уравнение делится на две части — та, в которую указанные переменные входят в первой степени и та, в которую входят все остальные члены уравнения. Матрица формируется из первой части. Рассмотрим пример ее работы:

```
> eqns := [x1+x2+x3 = 0, x1^2+x2+2 = 1, x1+x2-x3 = 3]:

> vars := [x1,x2]:

> GenerateMatrix(eqns, vars)
```

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} -x3 \\ -x1^2 - 1 \\ x3 + 3 \end{bmatrix}$$

Основная процедура данного раздела ODESYSconvert имеет следующий прототип:

```
ODESYSconvert(eqns).
```

Ее единственным входным параметром является eqns — система обыкновенных дифференциальных уравнений произвольного порядка, заданная списком или множеством обыкновенных дифференциальных уравнений.

Результатом ее работы является список матриц $A = [A_0, A_1, \ldots, A_{r-1}, A_r]$, задающих исходную систему в матричном виде (2).

Пример работы ODESYSconvert:

> v1 := $x^2 * \frac{d^2}{dx^2}$y1(x) + x*$\frac{d}{dx}$y1(x) - 2*$x^2 * \frac{d}{dx}$y2 - y1(x) - 2*x^2*y2(x) = 0:

> v2 := x*$\frac{d}{dx}$y2(x) + y2(x) = 0:

> ODESYSconvert([v1, v2])

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

3.2 Процедура ODESYSordreduce

Для преобразования системы произвольного порядка, заданной в матричном виде (2) в систему первого порядка была реализована роцедура ODESYSordreduce. Реализация была выполнена на основе описанного в разделе 2.1 способа перехода от системы произвольного порядка к системе первого порядка. Ниже приводится текст данной процедуры.

```
ODESYSordreduce := proc(A::list)
    local r, m, A1, A0;

    if nops(A) < 2 then
        error "can't be converted to first order differential system";
    elif nops(A) = 2 then
        return A[2], A[1];
    end if;

    r := nops(A)-1;
    m := op(A[1])[1];

    # construct A1
    A1 := Matrix(r*m, r*m, (i,j)->`if`(i=j, 1, 0));
    # assign bottom right m*m submatrix to A_r
    A1[(r-1)*m+1 .. r*m, (r-1)*m+1 .. r*m] := A[-1];

    # construct A0
    A0 := < Matrix((r-1)*m, r*m, (i,j)->`if`(j=i+m, -1, 0)), Matrix(A[1..-2]) >;

    return A1, A0;
end proc;
```

Ее входной параметр A есть список матричных коэффициентов, задающих дифференциальную систему произвольного порядка в матричном виде. На выход процедура подает матрицы A_1 и A_0, задающие дифференциальную систему первого порядка вида (1).

3.3 Процедура ExtrAB

Алгоритм ExtrAB, описанный в разделе 2.4, был реализован в Maple в виде процедуры ExtrAB с учетом указанных ранее требований к реализации. На вход этой процедуре подается дифференциальная система eqns в виде множества или списка обыкновенных дифференциальных уравнений, а также множество ns выделенных неизвестных. Предполагается, что в системе присутствует только одна независимая переменная.

Внутри в первую очередь с помощью вспомогательной процедуры ODESYSconvert, реализованной как часть реализации ExtrAB, список уравнений eqns, задающий исходную систему, преобразуется в список матриц mxs (матричные коэффициенты системы).

Затем, с помощью еще одной вспомогательной процедуры ODESYSordreduce, так же реализованной как часть реализации ExtrAB, полученный на предыдущем шаге список матриц mxs, задающий исходную систему произвольного порядка, преобразуется в две матрицы A1 и A0, задающие систему первого порядка, к которой таким образом была сведена исходная.

Для работы процедур Extract и ReducedSystem если не был передан параметр R, то производится подготовительный шаг — для R фиксируется значение 'differential'.

Затем, производится проверка, является ли ведущая матрица A1 системы обратимой. Если ведущая матрица обратима, то имеющаяся система первого порядка легко сводится к нормальной системе и легко находится матрица M этой системы. Множество выделенных неизвестных ns остается неизменным и копируется в ns1.

Если ведущая матрица не является обратимой, то для исходной системы производится вызов процедуры Extract, результатом работы которой является матрица M результирующей нормальной системы, а также списка соответствий между индексами неизвестных в исходной системе и в результирующей. Так же результатом работы Extract является матрица T1 линейного преобразования, с помощью которой выделенные неизвестные, не вошедшие в новую нормальную дифференциальную систему, могут быть выражены через выделенные неизвестные, вошедшие в новую систему.

В случае, если все выделенные неизвестные вошли в полученную нормальную систему, матрица преобразования T1 отсутствует.

В случае применения процедуры Extract, дополнительно, ввиду возможного уменьшения количества оставшихся в системе выделенных неизвестных, а также возможной смены их индексов соответствующим образом изменяется и заполняется текущее множество выделенных неизвестных ns1.

Полученные тем или иным способом матрица нормальной системы M и текущий список выделенных неизвестных ns1 далее подаются на вход процедуре ReducedSystem, реализующей AB-алгоритм.

Результатом работы ReducedSystem является матрица финальной нормальной системы B, а также список соответствия индексов.

В заключение, имея финальную нормальную систему, задающуюся матрицей B, а также все необходимые соответствия индексов производится преобразование матрицы B в список уравнений A, представляющих собой линейные комбинации функций с соответствующими именами и возможно производных.

В случае, если при применении процедуры Extract часть выделенных неизвестных не вошла в новую систему, то по полученной в результате работы Extract матрице линейного преобразования T1 строится список уравнений T, задающих линейное выражение невошедших в результирующую систему выделенных неизвестных через вошедшие.

Процедура ExtrAB имеет следующий прототип:

```
ExtrAB(eqns, ns, R),
```

где eqns — линейная однородная система обыкновенных дифференциальных уравнений произвольного порядка, заданная списком или множеством дифференциальных уравнений, ns — множество выделенных неизвестных функций, заданных своими именами с указанием зависимости от независимой переменной, R — кольцо полиномов Оре.

Если параметр R отсутствует, то его значением по умолчанию считается 'differential'.

Результатом работы процедуры являются:

- `A` — список дифференциальных уравнений, задающий результирующую систему;
- `T` — список алгебраических уравнений, задающих линейные выражения не вошедших в результирующую систему выделенных неизвестных через вошедшие.

В случае, если при применении процедуры `Extract` все выделенные неизвестные попали в результирующую систему, параметр `T` отсутствует.

Рассмотрим несколько примеров данной процедуры.

Пример 3. На вход подается система, которая в матричной форме имеет вид

$$\begin{bmatrix} -1 & 0 & -x & 0 & x & 0 \\ 0 & 0 & x(x+1) & 0 & 0 & x \\ 0 & 0 & x & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} y' + \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & -(x+1) \\ 0 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 1 & 0 & 1 \\ 1 & x & 0 & 0 & 0 & 0 \end{bmatrix} y = 0$$

Ее вектор неизвестных состоит из функций `y1`,`y2`,`y3`,`y4`,`y5` и `y6`. Пусть `y1`,`y2` — выделены.

```
> eqns := [-diff(y1(x),x)-x*(diff(y3(x),x))+y4(x)+x*(diff(y5(x),x))+y5(x)+y6(x)=0,
          (x+1)*x*(diff(y3(x), x))-y4(x)+x*(diff(y6(x), x))+(-x-1)*y6(x)=0,
          x*(diff(y3(x), x))+diff(y4(x), x)-y4(x)+diff(y6(x), x)-y6(x)=0,
          diff(y3(x), x)-y5(x)=0,
          -y1(x)+y4(x)+y6(x)=0,
          y1(x)+x*y2(x)=0]

> ExtrAB(eqns, {y1(x),y2(x)})
```

$$\left[\frac{d^2}{dx^2}y2(x) = \frac{1}{x}y2(x) + \left(1 - \frac{2}{x}\right)\frac{d}{dx}y2(x)\right], \left[y1 = -xy2\right]$$

Пример 4. На вход подается система, которая в матричной форме имеет вид

$$\begin{bmatrix} x^2 & 0 \\ 0 & 0 \end{bmatrix} y'' + \begin{bmatrix} x & -2x^3 \\ 0 & x \end{bmatrix} y' + \begin{bmatrix} -1 & -2x^2 \\ 0 & 1 \end{bmatrix} y = 0$$

Ее вектор неизвестных состоит из двух функций `y1` и `y2`. Пусть `y1` — выделена.

```
> eqns := [x^2(diff(y1(x),x,x))+x(diff(y1(x),x))-2x^3(diff(y2(x),x))-y1(x)-
          2x^2(y2(x))=0, x(diff(y2(x),x))+y2(x)=0]

> ExtrAB(eqns, {y1(x)})
```

$$\left[\frac{d^2}{dx^2}y1(x) = \frac{1}{x^2}y1(x) - \frac{1}{x}\frac{d}{dx}y1(x)\right]$$

В данном случае единственная выделенная неизвестная вошла в результирующую систему и поэтому алгебраическая система `T` отсутствует.

Заключение

В результате выполнения выпускной квалификационной работы был реализован в системе компьютерной алгебры `Maple` алгоритм `ExtrAB`, обобщающющий AB-алгоритм на случай произвольных линейных однородных дифференциальных систем полного ранга. Для этого:

- была реализована процедура `ODESYSconvert` преобразования дифференциальной системы, заданной в стандартном для `Maple` виде в матричный вид;
- была реализована процедура `ODESYSordreduce` сведения системы произвольного порядка, заданной в матричном виде, к системе первого порядка;
- была реализована на основе существующих реализаций алгоритма `Extract` и AB-алгоритма, а также вспомогательных процедур, основная процедура `ExtrAB`, принимающая на вход линейную однородную дифференциальную систему произвольного порядка, заданную в стандартном для `Maple` виде, и возвращающая новую нормальную систему, в которую не входят невыделенные неизвестные и которая задана в стандартном для `Maple` виде.

Список литературы

[1] Абрамов С. А., Бронштейн М. *Решение линейных дифференциальных и разностных систем по отношению к части неизвестных* // Журнал вычисл. матем. и матем. физ. — 2006. № 2. — с. 229–241.

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

[3] Panferov A. A. *Linear differential-algebraic systems with selected unknowns* [Электронный ресурс]. — Электрон. дан. — URL: indico.math.cnrs.fr/event/919/contribution/29/material/slides/0.pdf

[4] Bronstein M. , Petkovšek M. *On Ore rings, linear operators and factorisation* // Программирование. — 1994. № 1, — с. 27–44.

[5] Bronstein M. , Petkovšek M. *An introduction to pseudo-linear algebra* // Theoretical Computer Science. — 1996. V. 157. — p. 3–33

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