ISSN 0132-3474

Номер 2

Март - Апрель 2024

ПРОГРАММИРОВАНИЕ

НАУКА

1727

СОДЕРЖАНИЕ

Номер 2, 2024

КОМПЬЮТЕРНАЯ АЛГЕБРА

Семинар по компьютерной алгебре в 2022–2023 гг.
С. А. Абрамов, А. А. Боголюбская 3

Компьютерно-алгебраический подход к построению первых дифференциальных приближений: осциллятор Ван дер Поля
Ю. А. Блинков 7

Расширяемое эссе о системе компьютерной алгебры Sage и редактор для создания расширяемых эссе
Е. А. Бордаченкова, В. Н. Зубарева, А. А. Панфёров 13

Интегрирование вырожденной системы ОДУ
А. Д. Брюно, В. Ф. Еднерал 22

Построение компартментальных моделей динамических систем с применением программного комплекса символьных вычислений на языке Julia
А. В. Демидова, О. В. Дружинина, О. Н. Масина, А. А. Петров 33

Символьно-численная реализация модели адиабатических волноводных мод для двумерных нерегулярных волноводов
Д. В. Диваков, А. А. Тютюнник, Д. А. Стариков 45

Реализация аналитической проективной геометрии для компьютерной графики
М. Н. Геворкян, А. В. Королькова, Д. С. Кулябов, Л. А. Севастьянов 51

Символьные исследования уравнений Максвелла в формализме пространственно-временной алгебры
А. В. Королькова, М. Н. Геворкян, А. В. Фёдоров, К. А. Штепа, Д. С. Кулябов 66

О вычислении частичных сумм кратных числовых рядов методами компьютерной алгебры
В. И. Кузоватов, А. А. Кытманов, Е. К. Мышкина 74

Применение систем компьютерной алгебры для исследования тождеств Чаунди–Булларда для функции векторного разбиения с весом
А. Б. Лейнартене, А. П. Ляпин 79

Примитивные элементы свободных неассоциативных алгебр над конечными полями
М. В. Майсурадзе, А. А. Михалёв 84

Порт-Гамильтоновы системы: распознавание структуры и приложения
В. Н. Сальников 93

Нижние границы для ранга матрицы с нулями и единицами вне главной диагонали
А. В. Селиверстов, О. А. Зверков 100

Поиск лорановых решений усеченных систем линейных дифференциальных уравнений с использованием EG-исключений
А. А. Рябенко, Д. Е. Хмельнов 108

Решение задач анализа райсовских данных: теория и численное моделирование методами компьютерной алгебры в системе Wolfram Mathematica
Т. В. Яковлева 118

CONTENTS

No. 2, 2024

COMPUTER ALGEBRA

Seminar on Computer Algebra in 2022–2023
S. A. Abramov, A. A. Bogolyubskaya 3

Computer-Algebraic Approach to First Differential Approximations: Van der Pol Oscillator
Yu. A. Blinkov 7

An Extendable Essay on the Sage Computer Algebra System and an Editor for Creating Extendable Essays
E. A. Bordachenkova, V. N. Zubareva, A. A. Panferov 13

Integration of a Degenerate System of ODEs
A. D. Bruno, V. F. Edneral 22

Constructing Compartmental Models of Dynamic Systems Using a Software Package for Symbolic Computation in Julia
A. V. Demidova, O. V. Druzhinina, O. N. Masina, A. A. Petrov 33

Symbolic-Numerical Implementation of the Model of Adiabatic Guided Modes for Two-Dimensional Irregular Waveguides
D. V. Divakov, A. A. Tyutyunnik, D. A. Starikov 45

Implementation of Analytic Projective Geometry for Computer Graphics
M. N. Gevorkyan, A. V. Korol'kova, D. S. Kulyabov, L. A. Sevast'yanov 51

Symbolic Studies of Maxwell's Equations in Space-Time Algebra Formalism
A. V. Korol'kova, M. N. Gevorkyan, A. V. Fedorov, K. A. Shtepa, D. S. Kulyabov 66

On Calculating Partial Sums of Multiple Numerical Series by Methods of Computer Algebra
V. I. Kuzovatov, A. A. Kytmanov, E .K. Myshkina 74

Applying Computer Algebra Systems to Study Chaundy-Bullard Identities for the Vector Partition Function with Weight
A. B. Leinartene, A. P. Lyapin 79

Primitive Elements of Free Non-Associative Algebras over Finite Fields
M. V. Maisuradze, A. A. Mikhalev 84

Port-Hamiltonian Systems: Structure Recognition and Applications
V. N. Salnikov 93

Lower Bounds for the Rank of a Matrix with Zeros and Ones outside the Leading Diagonal
A. V. Seliverstov, O. A. Zverkov 100

Searching for Laurent Solutions of Truncated Systems of Linear Differential Equations with the Use of EG-Eliminations
A. A. Ryabenko, D. E. Khmelnov 108

Solving Rician Data Analysis Problems: Theory and Numerical Modeling Using Computer Algebra Methods in Wolfram Mathematica
T. V. Yakovleva 118

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 3–6*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

СЕМИНАР ПО КОМПЬЮТЕРНОЙ АЛГЕБРЕ В 2022–2023 ГГ.

© 2024 г. С. А. Абрамов[a], *, А. А. Боголюбская[b], **

[a] *Федеральный исследовательский центр "Информатика и управление" РАН, Вычислительный центр РАН им. А.А. Дородницына 119333 Москва, ул. Вавилова, д. 40, Россия*

[b] *Объединенный институт ядерных исследований 141980 Дубна, Московская область, Россия*

**E-mail: sergeyabramov@mail.ru*
***E-mail: abogol@jinr.ru*

Поступила в редакцию 30.08.2023
После доработки 25.09.2023
Принята к публикации 25.09.2023

Годовой отчет о работе научно-исследовательского семинара по компьютерной алгебре.

Ключевые слова: научно-исследовательский семинар по компьютерной алгебре
DOI: 10.31857/S0132347424020012 **EDN:** RPFTBM

О СЕМИНАРЕ

На семинаре рассматриваются новые результаты в области компьютерной алгебры — символьные алгоритмы и их реализация, соответствующие вопросы системного программирования.

В 2022–2023 учебном году семинар собирался раз в месяц по третьим средам (онлайн).

РЕГУЛЯРНЫЕ СОБРАНИЯ СЕМИНАРА

С сентября по май были прочитаны следующие доклады[1]. Аннотации доступны на странице семинара, http://www.ccas.ru/sabramov/seminar/doku.php, где также содержится информация о состоявшихся ранее докладах.

А.Н. Прокопеня (Варшавский университет естественных наук — SGGW, Варшава, Польша; alexander_prokopenya@sggw.edu.pl)

Построение периодических решений уравнений движения машины Атвуда с двумя колеблющимися грузами.

С.А. Гутник (Московский государственный институт международных отношений МИД России, Московский физико-технический институт, Москва; sergey.gutnik@gmail.com), В.А.Сарычев (ИПМ им. М.В. Келдыша РАН, Москва; vas31@rambler.ru)

Исследование положений равновесия двух связанных тел на круговой орбите с применением методов компьютерной алгебры.

Г. Погудин (Политехническая школа, Париж, Франция; gleb.pogudin@polytechnique.edu)

Решение разностных уравнений в последовательностях: что можно и чего нельзя.

А.Б. Батхин (Институт прикладной математики им. М.В. Келдыша РАН, Московский физико-технический институт, Москва; batkhin@gmail.com), З.Х. Хайдаров (Самаркандский государственный университет им. Ш.Рашидова, Самарканд, Узбекистан; zafarxx@gmail.com)

Структура резонансных многообразий в многочастотных системах Гамильтона.

В.Ф. Еднерал (Научно-исследовательский институт ядерной физики имени Д.В. Скобельцына МГУ, Москва; edneral@theory.sinp.msu.ru)

Об интегрируемости автономных систем ОДУ с зависящей от параметров полиномиальной правой частью.

Е.В. Зима (Университет Уилфрида Лорие, Ватерлоо, Канада; ezima@wlu.ca)

О специальном выборе модулей в модулярной арифметике.

[1] Перечень докладов, прочитанных в 1995–2022 г.г., опубликован в [29]–[56].

И.Е. Широков (Физический факультет МГУ им. М.В. Ломоносова, Москва; shi95@yandex.ru)

Применение методов компьютерной алгебры к квантовым вычислениям в суперсимметричных теориях.

Т.В. Яковлева (Федеральный исследовательский центр «Информатика и управление» РАН, Москва; tan-ya@bk.ru)

Двухпараметрический анализ райсовских данных: основы теории и результаты численного моделирования с помощью системы Wolfram Mathematica.

К.Д. Царегородцев (Мехмат МГУ им. М.В. Ломоносова, Москва, АО «НПК «Криптонит», Москва; kirill94_12@mail.ru)

Правильные семейства дискретных функций: эквивалентные определения и свойства.

МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ “КОМПЬЮТЕРНАЯ АЛГЕБРА”

26–28 июня 2023 г. состоялась 5-я Международная конференция “Компьютерная алгебра”, организованная ФИЦ ИУ РАН, ИПМ РАН и РУДН. Материалы конференции доступны на сайте http://www.ccas.ru/ca/conference.

Одна из сессий конференции была посвящена памяти замечательного ученого и человека — Марко Петковшека (10.04.1955-24.03.2023). Последнее его выступление на нашем семинаре с докладом “Линейные рекуррентные уравнения с полиномиальными коэффициентами: метод факториального базиса для нахождения решений, имеющих вид определенных сумм” (совместным с А. Хименесом-Пастором) состоялось 20 апреля 2022 г., http://www.ccas.ru/sabramov/seminar/lib/exe/fetch.php?media = petkovsek220420.pdf .

БЛАГОДАРНОСТИ

Участники семинара и 5-й Международной конференции “Компьютерная алгебра” выражают благодарность факультету ВМК МГУ и в особенности Т.Е. Романенко – за содействие в организации онлайн-трансляций.

СПИСОК ЛИТЕРАТУРЫ

1. *Абрамов С.А., Зима Е.В.* Семинар по компьютерной алгебре на факультете вычислительной математики и кибернетики МГУ в 1995–1996 г. // Программирование. 1997. № 1. С. 75–77.
2. *Абрамов С.А., Зима Е.В.* Научно-исследовательский семинар «Компьютерная алгебра» в 1996–1997 г. // Программирование. 1998. № 1. С. 69–72.
3. *Абрамов С.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 1997–1998 г. // Программирование. 1998. № 6. С. 3–7.
4. *Абрамов С.А., Крюков А.П., Ростовцев В.А.* Семинар по компьютерной алгебре в 1998–1999 г. // Программирование. 2000. № 1. С. 8–12.
5. *Абрамов С.А., Крюков А.П., Ростовцев В.А.* Семинар по компьютерной алгебре в 1999–2000 г. // Программирование. 2001. № 1. С. 3–7.
6. *Абрамов С.А., Крюков А.П., Ростовцев В.А.* Семинар по компьютерной алгебре в 2000–2001 г. // Программирование. 2002. № 2. С. 6–9.
7. *Абрамов С.А., Крюков А.П., Ростовцев В.А.* Семинар по компьютерной алгебре в 2001–2002 г. // Программирование. 2003. № 2. С. 3–7.
8. *Абрамов С.А., Еднерал В.Ф., Ростовцев В.А.* Семинар по компьютерной алгебре в 2002–2003 г.// Программирование. 2004. № 2. С. 3–7.
9. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2003–2004 г. // Программирование. 2005. № 2. С. 3–9.
10. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2004–2005 г. // Программирование. 2006. № 2. С. 3–7.
11. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2005–2006 г. // Программирование. 2007. № 2. С. 3–8.
12. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2006–2007 г. // Программирование. 2008. № 2. С. 3–8.
13. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2007–2008 г. // Программирование. 2009. № 2. С. 3–9.
14. «Mathematical Modeling and Computational Physics (CAAP›2009)». Book of abstracts of the internationl conference. Dubna, July 7–11, 2009. Dubna, 2009.
15. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А., Еднерал В.Ф.* Семинар по компьютерной алгебре в 2008–2009 г. // Программирование. 2010. № 2. С. 3–8.
16. *Абрамов С.А., Боголюбская А.А., Еднерал В.Ф., Ростовцев В.А.* Семинар по компьютерной алгебре в 2009–2010 г. // Программирование. 2011. № 1. С. 3–8.
17. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 2010–2011 г. // Программирование. 2012. № 2. С. 3–8.

18. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 2011–2012 г. // Программирование. 2013. № 2. С. 3–10.
19. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 2012–2013 г. // Программирование. 2014. № 2. С. 3–11.
20. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 2013–2014 г. // Программирование. 2015. № 2. С. 3–6.
21. *Абрамов С.А., Боголюбская А.А., Ростовцев В.А.* Семинар по компьютерной алгебре в 2014–2015 г. // Программирование. 2016. № 2. С. 4–7.
22. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2015–2016 г. // Программирование. 2017. № 2. С. 3–6.
23. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2016–2017 г. // Программирование. 2018. № 2. С. 3–4.
24. [52] *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2017–2018 г. // Программирование. 2019. № 2. С. 3–5.
25. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2018–2019 г. // Программирование. 2020. № 2. С. 3–5.
26. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2019–2020 г. // Программирование. 2021. № 2. С. 3–4.
27. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2020–2021 г. // Программирование. 2022. № 2. С. 3–6.
28. *Абрамов С.А., Боголюбская А.А.* Семинар по компьютерной алгебре в 2021–2022 г. // Программирование. 2023. № 2. С. 3–4.

SEMINAR ON COMPUTER ALGEBRA IN 2022–2023

© 2024 S. A. Abramov[a], A. A. Bogolyubskaya[b]

[a] *Federal Research Center «Informatics and Management» RAS*
Computing Center RAS named after. A.A. Dorodnitsyna
st. Vavilova 40, Moscow, 119333 Russia

[b] *Joint Institute for Nuclear Research*
Dubna, Moscow region, 141980 Russia

Annual report on the work of the research seminar on computer algebra.

Keywords: research seminar on computer algebra

REFERENCES

1. *Abramov S.A., Zima E.V.* Seminar on computer algebra at the Faculty of Computational Mathematics and Cybernetics of Moscow State University in 1995–1996 // Programming and Computer Software. 1997. No. 1. P. 75–77.
2. *Abramov S.A., Zima E.V.* Scientific-research seminar “Computer Algebra” in 1996–1997 // Programming and Computer Software. 1998. No. 1. P. 69–72.
3. *Abramov S.A., Rostovtsev V.A.* Seminar on computer algebra in 1997–1998 // Programming and Computer Software. 1998. No. 6. P. 3–7.
4. *Abramov S.A., Kryukov A.P., Rostovtsev V.A.* Seminar on computer algebra in 1998–1999 // Programming and Computer Software. 2000. No. 1. P. 8–12.
5. *Abramov S.A., Kryukov A.P., Rostovtsev V.A.* Seminar on computer algebra in 1999–2000 // Programming and Computer Software. 2001. No. 1. P. 3–7.
6. *Abramov S.A., Kryukov A.P., Rostovtsev V.A.* Seminar on computer algebra in 2000–2001 // Programming and Computer Software. 2002. No. 2. P. 6–9.
7. *Abramov S.A., Kryukov A.P., Rostovtsev V.A.* Seminar on computer algebra in 2001–2002 // Programming and Computer Software. 2003. No. 2. P. 3–7.
8. *Abramov S.A., Edneral V.F., Rostovtsev V.A.* Seminar on computer algebra in 2002–2003 // Programming and Computer Software. 2004. No. 2. P. 3–7.
9. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2003–2004 // Programming and Computer Software. 2005. No. 2. P. 3–9.
10. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2004–2005 // Programming and Computer Software. 2006. No. 2. P. 3–7.
11. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2005–2006 // Programming and Computer Software. 2007. No. 2. P. 3–8.
12. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2006–2007 // Programming and Computer Software. 2008. No. 2. P. 3–8.

13. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2007–2008 // Programming and Computer Software. 2009. No. 2. P. 3–9.
14. "Mathematical Modeling and Computational Physics (CAAP'2009)". Book of abstracts of the international conference. Dubna, July 7–11, 2009. Dubna, 2009.
15. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A., Edneral V.F.* Seminar on computer algebra in 2008–2009 // Programming and Computer Software. 2010. No. 2. P. 3–8.
16. *Abramov S.A., Bogolyubskaya A.A., Edneral V.F., Rostovtsev V.A.* Seminar on computer algebra in 2009–2010 // Programming and Computer Software. 2011. No. 1. P. 3–8.
17. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A.* Seminar on computer algebra in 2010–2011 // Programming and Computer Software. 2012. No. 2. P. 3–8.
18. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A.* Seminar on computer algebra in 2011–2012 // Programming and Computer Software. 2013. No. 2. P. 3–10.
19. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A.* Seminar on computer algebra in 2012–2013 // Programming and Computer Software. 2014. No. 2. P. 3–11.
20. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A.* Seminar on computer algebra in 2013–2014 // Programming and Computer Software. 2015. No. 2. P. 3–6.
21. *Abramov S.A., Bogolyubskaya A.A., Rostovtsev V.A.* Seminar on computer algebra in 2014–2015 // Programming and Computer Software. 2016. No. 2. P. 4–7.
22. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2015–2016. Programming // Programming and Computer Software. No. 2. P. 3–6.
23. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2016–2017 // Programming and Computer Software. 2018. No. 2. P. 3–4.
24. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2017–2018 // Programming and Computer Software. 2019. No. 2. P. 3-5.
25. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2018–2019 // Programming and Computer Software. 2020. No. 2. P. 3–5.
26. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2019–2020 // Programming and Computer Software. 2021. No. 2. P. 3–4.
27. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2020–2021 // Programming and Computer Software. 2022. No. 2. P. 3–6.
28. *Abramov S.A., Bogolyubskaya A.A.* Seminar on computer algebra in 2021–2022 // Programming and Computer Software. 2023. No. 2. P. 3–4.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 7–12*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК. 681.3.06, 004.94, 519.62

КОМПЬЮТЕРНО-АЛГЕБРАИЧЕСКИЙ ПОДХОД К ПОСТРОЕНИЮ ПЕРВЫХ ДИФФЕРЕНЦИАЛЬНЫХ ПРИБЛИЖЕНИЙ: ОСЦИЛЛЯТОР ВАН ДЕР ПОЛЯ

© 2024 г. Ю. А. Блинков[a,*]

[a]*Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского 410012 Саратов, ул. Астраханская, д. 83, Россия*

**E-mail: blinkovua@info.sgu.ru*

Поступила в редакцию 01.08.2023
После доработки 10.09.2023
Принята к публикации 01.10.2023

На примере первого дифференциального приближения проведено исследование для различных численных методов решения систем обыкновенных дифференциальных уравнений. Это позволило оценить жесткость системы ОДУ, которая моделирует колебания осциллятора Ван дер Поля, невязку метода, и предложить простые критерии выбора шага при проведении расчетов. Представленные методы позволяют провести эффективные вычисления средствами систем компьютерной алгебры.

Ключевые слова: численные методы решения ОДУ, первое дифференциальное приближение, компьютерная алгебра, базисы Грёбнера

DOI: 10.31857/S0132347424020022 **EDN:** RPFLEU

1. ВВЕДЕНИЕ

В 60-х годах прошлого века Н.Н. Яненко [1] сформулировал метод дифференциальных приближений разностной схемы. Основная идея этого метода состоит в замене исследования свойств разностной схемы исследованием некоторой задачи с дифференциальными уравнениями, занимающими промежуточное положение между исходной дифференциальной задачей и аппроксимирующей ее разностной схемой. В работах Н.Н. Яненко и его учеников в результате были сформулированы такие понятия, как *аппроксимационная вязкость разностной схемы* и *первое дифференциальное приближение разностной схемы*.

Первое дифференциальное приближение (ПДП) для дифференциальных уравнений в частных производных эволюционного типа, в частности, уравнения Кортевега-де Фриза с использованием систем компьютерной алгебры рассмотрено в [16], а для уравнений Навье–Стокса в [17] .

В [18] рассмотрено ПДП для разностных схем, описывающих обыкновенные дифференциальные уравнения. Обсуждена связь между сингулярным возмущением исходной системы и понятием ПДП. Для этого простого случая показана связь между методом оценки ошибки аппроксимации решения, основанным на анализе ПДП, и методом Ричардсона–Калиткина. Приведены примеры, когда при аппроксимации совместной системы дифференциальных уравнений в частных производных не всегда получаются совместные разностные системы уравнений. В качестве способа проверки совместности системы разностных уравнений предлагается проверять совместность ПДП для разностной системы. Рассмотрены вопросы вычисления ПДП в системах компьютерной алгебры Sage и SymPy.

Существует большое количество численных методов решения систем обыкновенных дифференциальных уравнений (ОДУ). Применение для их исследования ПДП позволяет получить информацию о качестве выбранного численного метода для конкретной системы, используя только символьные вычисления. В данной работе будут рассмотрены методы Рунге–Кутты [19] и многошаговые методы Адамса–Башфорта и Адамса–Мультона [20, 21] на примере осциллятора Ван дер Поля [22].

2. ПДП ДЛЯ СИСТЕМ ОДУ

Для систем ОДУ, в отличие от общего случая дифференциальных уравнений, построение ПДП не требует применения специальных алгоритмов. Обычно для численных методов требуется система ОДУ первого порядка, разрешенная относительно первых производных. Алгоритм построения ПДП для такого вида систем представляет набор простых операций с формальными степенными рядами.

Рассмотрим систему ОДУ первого порядка, разрешенную относительно производных. Также наложим условие, что правая часть уравнений F_i представляет собой полиномы от искомых функций $u^1, \ldots u^n$:

$$\begin{cases} u_t^1 = F_1, \\ \dots \\ u_t^n = F_n. \end{cases} \tag{2.1}$$

Применяя к системе (2.1) конкретный численный метод и заменяя искомые функции их разложением в ряд Тейлора с шагом h, получим в любой выбранной точке набор уравнений, которые будут представлять собой равенство нулю степенных рядов по h с коэффициентами $G_{j,k}$ от искомых функций и их производных

$$\begin{cases} u_t^1 - F_1 + \sum\limits_{k=0}^{\infty} G_{1,k} h^k = 0, \\ \dots \\ u_t^n - F_n + \sum\limits_{k=0}^{\infty} G_{n,k} h^k = 0. \end{cases} \tag{2.2}$$

Система (2.2) образует дифференциальный идеал [23] относительно операций сложения, умножения и дифференцирования рядов, равных нулю. Сам вид рядов (2.2), разрешенных относительно первых производных, позволяет выполнить замену производных от функций $u^1,\dots u^n$ в коэффициентах $G_{j,k}$ через сами функции.

Можно сформулировать алгоритм построения ПДП, используя алгоритмы построения базисов Грёбнера [24] для рядов. Алгоритм построения ПДП оканчивается при получении первого ненулевого члена по степеням h, выраженного только через сами функции. Как следствие, вид ПДП не зависит от точки разложения в ряд Тейлора и представляет собой *канонический вид*.

На языке Python была написана программа с использованием системы компьютерной алгебры SymPy, которая строит ПДП для систем ОДУ, разрешенные относительно старших производных с полиномиальной правой частью. Головной модуль имеет имя fda_ode.py. Он и остальные файлы программ и файлы с приведенными ниже расчетами расположены по адресу https://github.com/blinkovua/sharing-blinkov/tree/master/FDA_ODE. Рассмотрим для примера уравнение Дуффинга [25], которое представляет собой важный пример системы (с одной степенью свободы) с нелинейной восстанавливающей силой $F(u) = u + \varepsilon u^3$ с малым параметром $\varepsilon \ll 1$.

$$\begin{cases} u_{tt} + u + \varepsilon u^3 = 0, \\ u_t^2 \,/\, 2 + u^2 \,/\, 2 + \varepsilon u^4 \,/\, 4 = C. \end{cases} \tag{2.3}$$

Уравнение (2.3) имеет первый интеграл с константой C.

Для применения методов Рунге–Кутты перепишем (2.3) в виде системы первого порядка

$$\begin{cases} u_t - u1 = 0, \\ u1_t + u + \varepsilon u^3 = 0, \\ u1^2/2 + u^2/2 + \varepsilon u^4/4 = C. \end{cases} \tag{2.4}$$

Пример использования приведен ниже.

```
1  from fda_ОДУ import *
2
3  eps, t, h = symbols(r'\varepsilon, t, h')
4  u, u1 = symbols('u, u1', cls=Function)
5
6  eq = u(t).diff(t, 2)  + u(t) + eps*u(t)**3
7  print(eq)
```

В строке 1 произведен импорт основного модуля с вызовом необходимых библиотек, включая и систему компьютерной алгебры SymPy. Далее в строках 2–3 и 6 описываются основные символьные переменные и задается само уравнение Дуффинга.

```
8  eq1 = (u(t).diff(t)**2/2 + u(t)**2/2 +\
9    eps*u(t)**4/4) - h
10 print(expand(eq1.diff(t) - eq*u(t).diff(t)))
```

В строках 8-10 производится проверка первого интеграла системы.

```
11  init((u(t), u1(t)), t, h)
12  set_clip(7, 6, Rational(0, 1))
```

На строке 11 функция init первым аргументом принимает список зависимых функций от второго аргумента t с шагом h. Функция init также определяет представление мономов в виде списка степеней, в данном случае порядок производной по времени, и номера зависимой функции, что стандартно при построении базисов Грёбнера [10].

Первый аргумент всегда должен быть больше второго, который обычно должен быть больше предполагаемого порядка численного метода, а последний влияет только на объем вычислений.

```
13 def RungeKutta4(f, y):
14     k1 = f(y)
15     k2 = f(y + h*k1/2)
16     k3 = f(y + h*k2/2)
17     k4 = f(y + h*k3)
18     return expand((k1 + 2*k2 + 2*k3 + k4)/6)
19
20 def f(y):
21     return Matrix([\
22         clip_n(y[1]),\
23         clip_n(-y[0] - eps*y[0]**3),\
24     ])
```

В строках 13-24 задан метод Рунге–Кутты [5] для автономных систем. Правая часть для системы ОДУ задается в строках 12-17 списком аргументов y[0], y[1] = u, u1. Для значительного ускорения громоздких символьных вычислений применяется функция clip_n, которая обрезает ряд до заданной степени h.

```
25  r = RungeKutta4(f, Matrix([u(t), u1(t)]))
26
27  F1 = clip((T(u(t+h))-T(u(t)))/h - T(r[0]))
28  prnlatex(F1, eps)
29
30  F2 = clip((T(u1(t+h))-T(u1(t)))/h - T(r[1]))
31  prnlatex(F2, eps)
```

В строках 25-31 происходит вычисление разложения метода Рунге-Кутты в ряд Тейлора в точке $t=0$

$$\begin{cases} -u_1 + u_t + h\left(\dfrac{\varepsilon u^3}{2} + \dfrac{u + u_{tt}}{2}\right)\ldots = 0 \\ \varepsilon u^3 + u + u1_t + h\left(\dfrac{3\varepsilon u^2 u_1}{2} + \dfrac{u_1 + u1_{tt}}{2}\right)\ldots = 0. \end{cases} \tag{2.5}$$

```
32  f1 = NF(F1, [u(t).diff(t), u1(t).diff(t)],\
33      [F1, F2], head=False)
34  prnlatex(f1, eps)
35
36  f2 = NF(F2, [u(t).diff(t), u1(t).diff(t)],\
37      [F1, F2], head=False)
38  prnlatex(f2, eps)
```

В строках 32–38 происходит приведение ПДП к каноническому виду, который, в частности, дает точный порядок численного метода:

$$\begin{cases} u_t - u_1 + h^4\left(\dfrac{\varepsilon^2 u^4 u_1}{10} + \right. \\ \left. + \dfrac{\varepsilon u_1\left(3u - u_1\right)\left(3u + u_1\right)}{120} + \dfrac{u_1}{120}\right) + \mathcal{O}(h^5) = 0, \\ u1_t + u + \varepsilon u^3 + h^4\left(-\dfrac{3\varepsilon^3 u^7}{80} - \right. \\ \left. - \ldots - \dfrac{\varepsilon u\left(5u^2 + 18u_1^2\right)}{240} - \dfrac{u}{120}\right) + \mathcal{O}(h^5) = 0. \end{cases} \tag{2.6}$$

Первый аргумент содержит обрезанный ряд исходного разложения, второй список производных, которые являются старшими относительно операций дифференцирования в первых членах разложения, заданных в третьем аргументе в виде списка. Явное задание выбрано из нелинейности уравнений и сложного алгоритма определения старшего члена. Это обусловлено тем, что старший член может входить в несколько членов полинома от функций и их производных. Поэтому для проведения редукции необходимо выделить старший член сразу во всех частях полинома.

Параметр head=False показывает, что не нужно редуцировать первый член разложения. При его значении True будет происходить, если возможно, редукция головного члена. Это необходимо в том случае, если необходимо посчитать S-полином, как, например, сделано в работе [17] или посчитать первый интеграл, как показано ниже.

```
39  F3 = clip((T(u1(t+h))**2/2 +\
40    T(u(t+h))**2/2 + eps*T(u(t+h))**4/4) -\
41    (T(u1(t))**2/2 + T(u(t))**2/2 +\
42    eps*T(u(t))**4/4))
43  prnlatex(F3, eps)
44
45  f3 = NF(F3, [u(t).diff(t), u1(t).diff(t)],\
46    [f1, f2], head=True)
47  prnlatex(f3, eps)
```

В строках 39–43 происходит вычисление разложения первого интеграла из системы (2.4) в ряд Тейлора в точке $t=0$.

$$\begin{gathered} \varepsilon u^3 u_t + u u_t + u_1 u1_t + h\times \\ \times\left(\begin{array}{c} \dfrac{\varepsilon u^2\left(u u_{tt} + 3u_t^2\right)}{2} + \\ + \dfrac{u u_{tt} + u_1 u1_{tt} + u1_t^2 + u_t^2}{2} \end{array}\right)\ldots = 0. \end{gathered} \tag{2.7}$$

В строках 45–47 происходит приведение первого интеграла к каноническому виду. ПДП первого интеграла позволяет осуществлять дополнительный контроль численного метода для конкретного вида ОДУ (2.8).

$$h^4\left(\begin{array}{c} -\dfrac{\varepsilon^3 u^7 u_1}{16} - \dfrac{\varepsilon^2 u^3 u_1\left(3u^2 - 5u_1^2\right)}{24} - \\ - \dfrac{\varepsilon u u_1\left(3u^2 - 4u_1^2\right)}{48} \end{array}\right) + \mathcal{O}\left(h^5\right) = 0. \tag{2.8}$$

3. ПДП ДЛЯ ОСЦИЛЛЯТОРА ВАН ДЕР ПОЛЯ

Запишем уравнение осциллятора Ван дер Поля

$$u_{tt} - \mu\left(1 - u^2\right)u_t + u = 0 \tag{3.1}$$

в виде системы двух уравнений

$$\begin{cases} u_t - u_1 = 0, \\ u_{1t} - \mu\left(1 - u^2\right)u_1 + u = 0. \end{cases} \tag{3.2}$$

Применяя метод Рунге–Кутты четвертого порядка для (3.2), получим

$$\begin{cases} u_1 + h\left(-\dfrac{\mu u_1 (u-1)(u+1)}{2} - \dfrac{u}{2}\right) + \ldots = 0, \\ -\mu u_1 (u-1)(u+1) - u + \\ + h\left(\dfrac{\mu^2 u_1 (u-1)^2 (u+1)^2}{2} + \right. \\ \left. + \dfrac{\mu u\left(u^2 - 2u_1^2 - 1\right)}{2} - \dfrac{u_1}{2}\right) + \ldots = 0. \end{cases} \tag{3.3}$$

Разложение в ряд Тейлора при $t = 0$ для (3.3) дает

$$\begin{cases} u_t - u_1 + h\left(\dfrac{\mu u_1 (u-1)(u+1)}{2} + \right. \\ \left. + \dfrac{u + u_{tt}}{2}\right) + \mathcal{O}(h^2) = 0, \\ u_{1t} - \mu u_1\left(1 - u^2\right) + u + \\ + h\left(-\dfrac{\mu^2 u_1 (u-1)^2 (u+1)^2}{2} - \right. \\ \left. - \dfrac{\mu u\left(u^2 - 2u_1^2 - 1\right)}{2} + \dfrac{u_1 + u1_{tt}}{2}\right) + \mathcal{O}\left(h^2\right) = 0. \end{cases} \tag{3.4}$$

Используя алгоритмы построения базисов Грёбнера для рядов в системе компьютерной алгебры SymPy, построим ПДП для (3.4), в результате получим

$$\begin{cases} u_t - u_1 + h^4\left(\dfrac{\mu^4 u_1 (u-1)^4 (u+1)^4}{120} + \right. \\ \left. + \ldots\right) + \mathcal{O}\left(h^5\right) = 0, \\ u_{1t} - \mu u_1\left(1 - u^2\right) + u + \\ + h^4\left(-\dfrac{\mu^5 u_1 (u-1)^5 (u+1)^5}{120} - \right. \\ \left. - \ldots\right) + \mathcal{O}\left(h^5\right) = 0. \end{cases} \tag{3.5}$$

Построение ПДП позволило правильно определить порядок численного метода и его остаточный член. Были построены ПДП для различных явных и неявных Рунге–Кутты и многошаговых методов Адамса–Башфорта и Адамса–Мультона. Все результаты показали, что ПДП второго уравнения системы (3.2) имеет вид $h^p(C\mu^{p+1}u_1(u^2-1)^{p+1}) + \ldots)$, где p – порядок метода, C – некоторая константа. Значения констант C представлены в табл. 1.

Таблица 1

Метод	p	C
Гаусса–Лежандра (неявное правило средней точки)	2	1/12
Кранка–Николса [27]	2	1/12
Рунге–Кутты	4	–1/120
Адамса–Башфорта	4	–251/720
Адамса–Мультона	4	19/720
Дормана–Принса [28]	5	–1/3600
Адамса–Башфорта	5	95/288
Адамса–Мультона	5	19/720

Вычисление всех представленных примеров на компьютере с процессором Intel(R) Core(TM) i7-4770K CPU@3.50GHz в системе Debian 12 в оболочке Jupiter потребовало 620 сек и 160 Мегабайт оперативной памяти.

Остаточный член показывает жесткость системы [26] относительно параметра μ. В результате необходимо выбирать шаг h таким образом, чтобы обеспечить малость остаточного члена. Вычисленный остаточный член для методов Рунге–Кутты позволяет организовать эффективное определение размера шага h для заданной погрешности.

4. ЧИСЛЕННЫЕ ЭКСПЕРИМЕНТЫ

Были проведены расчеты для всех представленных примеров с начальными условиями $u = 0$, $u_t = 1$ при $t = 0$ до $t = 100$. Обозначение *max* показывает максимальное отклонение значения u от 0 и выбрано для контроля точности. Для табл. 2 шаг задан следующим значением $h = 0.01$. Символ * обозначает прерывание вычислений при значении $abs(u) >= 3$.

Как видно из табл. 2, значения $h^p\mu^{p+1}$ может использоваться для выбора h при вычислениях с фиксированным шагом. Также заметно преимущество неявных методов, по сравнению с явными методами, что возможно обусловлено меньшим значением константы C в ПДП.

Для вычислений с переменным шагом для заданной погрешности $tol = 0.01$ по следующей формуле $\min(h, (tol\ /\ (1 + abs(\text{остаточный член}))^{1/p})$. Как показано в табл. 3, при начальном шаге $h = 0.01$ все вычисления показали устойчивость при изменении шага и большую сходимость при большем порядке метода.

Таблица 2

μ	$h^p\mu^{p+1}$	max	μ	$h^p\mu^{p+1}$	max
Гаусса–Лежандра 2			Кранка–Николса 2		
0	0.00e+00	1.000	0	0.00e+00	1.000
1	1.00e-04	2.009	1	1.00e-04	2.009
5	1.25e-02	2.022	5	1.25e-02	2.022
10	1.00e-01	2.015	10	1.00e-01	2.015
20	8.00e-01	2.011	20	8.00e-01	2.011
50	1.25e+01	2.018	50	1.25e+01	2.018
100	1.00e+02	*	100	1.00e+02	*
150	3.38e+02	*	150	3.38e+02	*
Рунге–Кутты 4			Дормана–Принса 5		
0	0.00e+00	1.000	0	0.00e+00	1.000
1	1.00e-08	2.009	1	1.00e-10	2.009
5	3.13e-05	2.022	5	1.56e-06	2.022
10	1.00e-03	2.014	10	1.00e-04	2.014
20	3.20e-02	2.008	20	6.40e-03	2.008
50	3.12e+00	2.004	50	1.56e+00	2.001
100	1.00e+02	*	100	1.00e+02	2.265
150	7.59e+02	*	150	1.14e+03	*
Адамса–Башфорта 4			Адамса–Мультона 4		
0	0.00e+00	1.000	0	0.00e+00	1.000
1	1.00e-08	2.009	1	1.00e-08	2.009
5	3.13e-05	2.022	5	3.13e-05	2.022
10	1.00e-03	2.014	10	1.00e-03	2.014
20	3.20e-02	*	20	3.20e-02	2.008
50	3.12e+00	*	50	3.12e+00	2.008
100	1.00e+02	*	100	1.00e+02	*
150	7.59e+02	*	150	7.59e+02	*
Адамса–Башфорта 5			Адамса–Мультона 5		
0	0.00e+00	1.000	0	0.00e+00	1.000
1	1.00e-10	2.009	1	1.00e-10	2.009
5	1.56e-06	2.022	5	1.56e-06	2.022
10	1.00e-04	*	10	1.00e-04	2.014
20	6.40e-03	*	20	6.40e-03	2.008
50	1.56e+00	*	50	1.56e+00	2.022
100	1.00e+02	*	100	1.00e+02	*
150	1.14e+03	*	150	1.14e+03	*

5. ЗАКЛЮЧЕНИЕ

Проведенные вычисления показывают, что, применяя ПДП, можно оценить невязку используемого численного метода в зависимости от параметров задачи.

Применяя ПДП, возможно обнаружить и оценить жесткость системы ОДУ, а также использовать остаточный член ПДП для выбора постоянного шага или для управления переменным шагом.

Исходные тексты программ и представленные вычисления приведены по адресу github.com/blinkovua/sharing-blinkov/tree/master/FDA_ODE .

Представленные методы позволяют провести эффективные вычисления средствами систем компьютерной алгебры для решений систем ОДУ с правой полиномиальной частью.

Таблица 3

μ	шагов	max	μ	шагов	max
Гаусса–Лежандра 2			Кранка–Николса 2		
0	10001	1.000	0	10001	1.000
1	10001	2.009	1	10001	2.009
5	12999	2.022	5	14079	2.022
10	14922	2.014	10	16211	2.014
20	16783	2.008	20	18441	2.008
50	16899	2.003	50	18471	2.003
100	20332	2.001	100	22327	2.001
150	32912	2.001	150	35752	2.001
Рунге–Кутты 4			Дормана–Принса 5		
0	10001	1.000	0	10001	1.000
1	10001	2.009	1	10001	2.009
5	10001	2.022	5	10001	2.022
10	10024	2.014	10	10001	2.014
20	10179	2.008	20	10004	2.008
50	10225	2.003	50	10041	2.003
100	10273	2.001	100	10062	2.001
150	11796	2.001	150	10815	2.001

СПИСОК ЛИТЕРАТУРЫ

1. *Яненко Н.Н., Шокин Ю.И.* О первом дифференциальном приближении разностных схем для гиперболических систем уравнений // Сиб. матем. журн. 1969. Т. 10. № 5. С. 1173–1187. DOI: 10.1007/BF00971662
2. *Блинков Ю.А., Гердт В.П., Маринов К.Б.* Дискретизация квазилинейных эволюционных уравнений методами компьютерной алгебры // Программирование. 2017. № 2. С. 28–34.
3. *Блинков Ю.А., Ребрина А.Ю.* Исследования разностных схем для двумерных уравнений Навье–Стокса алгоритмами компьютерной алгебры // Программирование. 2023. № 1. С. 32–37.
4. *Блинкова А.Ю., Малых М.Д., Севастьянов Л.А.* О дифференциальных приближениях разностных схем // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2021. Т. 21. № 4. С. 472–488. DOI: 10.18500/1816-9791-2021-21-4-472-488
5. *Kutta M.* Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Zeitschrift für Mathematik und Physik. 1901. Vol. 46. P. 435–453.
6. *Bashforth F.* An Attempt to test the Theories of Capillary Action by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops, by J. C. Adams, Cambridge. 1883.
7. *Moulton F. R.* New methods in exterior ballistics, University of Chicago Press. 1926.

8. *van der Pol B.* On "Relaxation-Oscillations". The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1926. N. 2. P. 978–89992. DOI: 10.1080/14786442608564127
9. *Kolchin E. R.* Differential Algebra and Algebraic Groups. Academic Press, New York, 1973. 446 P.
10. *Buchberger B.* Gröbner bases: an Buchberger algorithmic method in polynomial ideal theory. Recent Trends in Multidimensional System Theory / Ed. by N. K. Bose. V. 6. 1985. P. 184–232.
11. *Duffing G.* Brzwungene Schwingungen bei veranderlicher Eigenfrequenz und ihre technische Bedeutung, Braunschweig, 1918.
12. *Curtiss C.F., Hirschfelder J.O.* Integration of stiff equations. Proceedings of the National Academy of Sciences of the USA. 1952. V. 38. N. 3. P. 235–243. DOI: 10.1073/pnas.38.3.235
13. *Crank J., Nicolson P.* A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. Proc. Camb. Phil. Soc. 1947. V. 43. N. 1. P. 50–67. DOI:10.1017/S0305004100023197
14. *Dorman J.R., Prince P.J.* A family of embedded Runge-Kutta formulae. Journal of Computational and Applied Mathematics. 1980. V. 6. N. 1. P. 19–26. DOI: 10.1016/0771-050X(80)90013-3

COMPUTER-ALGEBRAIC APPROACH TO FIRST DIFFERENTIAL APPROXIMATION: VAN DER POL OSCILLATOR

© 2024 Yu. A. Blinkov[a]

[a]*Saratov State National Research University, ul. Astrakhanskaya 83, Saratov, 410012 Russia*

First differential approximation has been used to analyze various numerical methods for solving systems of ordinary differential equations. This has made it possible to estimate the stiffness of the ODE system that models the oscillations of the Van der Pol oscillator and the error of the method as well as to propose simple criteria for choosing a calculation step. The presented methods allow one to perform efficient calculations using computer algebra systems.

Keywords: numerical methods for solving ODEs, first differential approximation, computer algebra, Gröbner bases

REFERENCES

1. *Yanenko N.N., Shokin Yu.I.* On the first differential approximation of difference schemes for hyperbolic systems of equations, Sib. Mat. Zh., 1969. V. 10. No. 5. P. 1173–1187.
2. *Blinkov Yu.A., Gerdt V.P., Marinov K.B.* Discretization of quasilinear evolution equations by computer algebra methods. Program. Comput. Software. 2017. V. 43. No. 2. P. 84–89.
3. *Blinkov Yu.A., Rebrina A.Yu.* Investigation of difference schemes for two-dimensional Navier–Stokes equations by using computer algebra algorithms, Program. Comput. Software, 2023. V. 49. No. 1. P. 26–31.
4. *Blinkova A.Yu., Malykh M.D., evast'yanov L.A.* On differential approximations of difference schemes, Izv. Sarat. Univ., Nov. Ser., Ser. Mat., Mekh., Inf. 2021. V. 21. No. 4. P. 472–488.
5. *Kutta M.* Beitrag zur näherungsweisen Integration totaler Differentialgleichungen. Zeitschrift für Mathematik und Physik. 1901. V. 46. P. 435–453.
6. *Bashforth F.* An Attempt to test the Theories of Capillary Action by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops, by J. C. Adams, Cambridge. 1883.
7. *Moulton F.R.* New methods in exterior ballistics, University of Chicago Press. 1926.
8. *van der Pol B.* On "Relaxation-Oscillations". The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. 1926. N. 2. P. 978–89992. DOI: 10.1080/14786442608564127
9. *Kolchin E.R.* Differential Algebra and Algebraic Groups. Academic Press, New York, 1973. 446 p.
10. *Buchberger B.* Gröbner bases: an Buchberger algorithmic method in polynomial ideal theory. Recent Trends in Multidimensional System Theory / Ed. by N. K. Bose. V. 6. 1985. P. 184–232.
11. *Duffing G.* Brzwungene Schwingungen bei veranderlicher Eigenfrequenz und ihre technische Bedeutung, Braunschweig, 1918.
12. *Curtiss C.F., Hirschfelder J.O.* Integration of stiff equations. Proceedings of the National Academy of Sciences of the USA. 1952. V. 38. N. 3. P. 235–243. DOI: 10.1073/pnas.38.3.235
13. *Crank J., Nicolson P.* A practical method for numerical evaluation of solutions of partial differential equations of the heat conduction type. Proc. Camb. Phil. Soc. 1947. V. 43. N. 1. P. 50–67. DOI:10.1017/S0305004100023197
14. *Dorman J.R., Prince P.J.* A family of embedded Runge–Kutta formulae. Journal of Computational and Applied Mathematics. 1980. V. 6. N. 1. P. 19–26. DOI: 10.1016/0771-050X(80)90013-3

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 13–21*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.5+004.9

РАСШИРЯЕМОЕ ЭССЕ О СИСТЕМЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ SAGE И РЕДАКТОР ДЛЯ СОЗДАНИЯ РАСШИРЯЕМЫХ ЭССЕ

© 2024 г. Е. А. Бордаченкова[a], *, В. Н. Зубарева[a], **, А. А. Панфёров[a], ***

[a]Московский государственный университет имени М. В. Ломоносова,
факультет высшей математики и кибернетики
119991 Москва, Ленинские горы, д. 1, стр. 52, Россия

**E-mail: lenabord@mail.ru*
***E-mail: zubareva.vita@mail.ru*
****E-mail: ast.a_s@mail.ru*

Поступила в редакцию 01.09.2023
После доработки: 10.09.2023
Принята к публикации 01.10.2023

Расширяемое эссе — особый формат электронных текстов, более удобный для чтения, чем гипертексты. Для облегчения создания и редактирования расширяемых эссе предлагается программа-редактор, реализованная в виде веб-приложения. С использованием программы-редактора написано расширяемое эссе о системе компьютерной алгебры Sage. Система Sage представляется удобной для получения первого опыта в области компьютерной алгебры для людей, впервые встречающихся с подобными системами.

Ключевые слова: расширяемое эссе, электронный текст, редактор эссе, Sage

DOI: 10.31857/S0132347424020038 **EDN:** RPFFQI

1. ВВЕДЕНИЕ

В современном мире людям чаще приходится иметь дело с электронными текстами, чем с текстами, напечатанными на бумаге. Количество электронных текстов лавинообразно растет.

С 1980-х годов проводятся работы, посвященные анализу различий физиологических и когнитивных процессов, происходящих при чтении бумажных и при чтении электронных текстов. Исследователи интересуются разными аспектами электронных текстов, анализируют влияние на читающего физических характеристик текста (таких как яркость и цвет экрана, расположение текста, начертание и размер шрифта и т.п.) и влияние структурных свойств электронных текстов (членение текста на логические фрагменты, размер отображаемых фрагментов). Не по всем вопросам ученые пришли к однозначным ответам, однако, некоторые выводы признаются научным сообществом. В обзоре [1] указывается: “Электронный текст как объект чтения действительно обладает рядом особенностей, из-за которых меняются психологические процессы, задействованные в чтении”. В некоторых работах выясняется, что у читателей при чтении электронных текстов по сравнению с чтением бумажных текстов происходит снижение скорости чтения, повышение когнитивной нагрузки, снижение точности (внимательности) чтения (см. обзор [1]).

Следует отметить, что процессы, задействованные при чтении линейного электронного текста, отличаются от процессов, происходящих при чтении гипертекста. В работе [2] указывается, что наличие ссылок увеличивает умственную нагрузку читающего двояко: напрямую, когда читатель принимает решение о переходе по ссылке, и на более высоком уровне понимания текста — переходы по ссылкам приводят к разрыву связи между частями текста, потере целостности восприятия текста. Форма организации электронного текста, описываемая ниже, разрабатывалась с целью облегчить читателю сохранение целостности восприятия текста.

В статье [3] предложена схема организации текста в виде расширяемого эссе. Расширяемое эссе — текст, представляющий собой краткое изложение материала (учебного, научного, познавательного) с опциональными расширениями, дополняющими исходный текст пояснениями, иллюстрациями, примерами и библиографическими ссылками. Для читателя первоначально расширяемое эссе выглядит как обычный текст, с традиционным разбиением на разделы и абзацы. Некоторые абзацы помечены специальными значками, показывающими, что абзац может быть “расширен”. Если читатель нажмет на значок расширения, абзац будет заменен другим, более пространным, абзацем или несколькими абзацами. Таким образом, читатель имеет возможность

настраивать текст, выбирать более краткий или более подробный способ изложения материала в зависимости от своих нужд. Принципиальное отличие расширяемого эссе от гипертекста состоит в том, что при любом расширении текст эссе сохраняет целостность и связность. Это позволяет надеяться, что изменение текста во время чтения не создаст неудобства читателю.

2. СТРУКТУРА РАСШИРЯЕМОГО ЭССЕ

Формат расширяемого эссе разрабатывался прежде всего как форма организации учебных материалов, отсюда происходят особенности этого формата [3].

Расширяемое эссе похоже на обычную книгу: имеет название, после которого указан автор, далее идет текст. Текст делится на разделы, которые состоят из абзацев. Каждый абзац может иметь расширение (более подробный вариант текста абзаца), в таком случае слева на полях абзаца стоит значок расширения. В конце раздела могут находиться вопросы для самопроверки.

Текст расширения структурно является абзацем или несколькими абзацами, каждый из которых также может иметь расширения (так сказать, расширения второго, третьего и т.д. уровней). Таким образом, расширяемое эссе имеет линейно-иерархическую структуру. Глубина вложенности расширений формально не ограничивается. Однако опыт показывает, что больше двух уровней расширений не требуется: эссе становится неудобно читать, поскольку появляется слишком сильный контраст в степени подробности изложения материала между основным, нерасширенным, текстом эссе и расширением максимальной глубины.

Расширения бывают нескольких видов, каждому из которых соответствует свой значок:

“+” — более подробное объяснение,

“:” — пример,

“[ ]” — ссылка на библиографический источник,

“T” — вопрос или задача по теме раздела.

Если читатель открыл расширение абзаца и решил, что краткого варианта абзаца ему достаточно для понимания, он может убрать (закрыть) расширение, вернувшись к первоначальному, краткому варианту абзаца, нажав значок “←” в левом поле расширения. На рис. 1 и 2 приведен пример фрагмента эссе с нераскрытым и с раскрытым расширением.

Расширение “T” отличается от других видов расширений, оно даёт возможность читателю протестировать себя. При нажатии на значок “T” читателю предъявляется вопрос и предлагается на выбор несколько вариантов ответов. После того как читатель указал выбранный ответ, ему сообщаются правильный ответ и пояснение к нему. Автор эссе может подготовить несколько вопросов к разделу. В таком случае вопрос, предъявляемый читателю, выбирается случайным образом из списка вопросов.

3. СОЗДАНИЕ РАСШИРЯЕМЫХ ЭССЕ

Поскольку формат расширяемого эссе предполагает разные уровни подробности изложения материала, автор должен определить, какие из фрагментов нужно объяснять более подробно (т. е. снабдить расширением). Далее, нужно написать не один полный текст по теме эссе; для некоторых фрагментов придется писать несколько вариантов — основной текст и расширение. Наконец, необходимо проверить, что текст эссе будет гладко читаться при любой комбинации фрагментов основного текста и расширений.

В работе [3] расширяемое эссе предлагается реализовать в виде HTML-файла, чтобы читатель мог открывать и читать расширяемое эссе с помощью своего интернет-браузера. Интерактивность (показ и скрытие расширений) обеспечивается программными вставками на языке JavaScript. Если эссе содержит иллюстрации (таблицы, рисунки), кроме HTML-файла, в реализацию эссе включаются соответствующие графические файлы. Реализация эссе хранится, например, у автора на сервере. Читатель обращается к серверу и просматривает эссе в режиме on-line либо скачивает все файлы, составляющие реализацию эссе, на свой компьютер и просматривает эссе без доступа к интернету.

Технически, согласно [3], процесс создания расширяемого эссе состоит в следующем. Первоначально текст эссе оформляется на языке XML; для задания иерархической структуры эссе определен набор XML-тегов. Затем специальная программа переводит текст в HTML-формат, вставляя программные фрагменты, обеспечивающие интерактивность эссе.

С использованием описанной технологии созданы три расширяемых эссе: “Сложность алгоритмов” ([4]), “Реализация расширяемого эссе” ([5]), “Портрет Адели Блох-Бауэр I” ([6]). Авторы столкнулись с проблемой ввода формул, поскольку формат XML не имеет средств для работы с формулами.

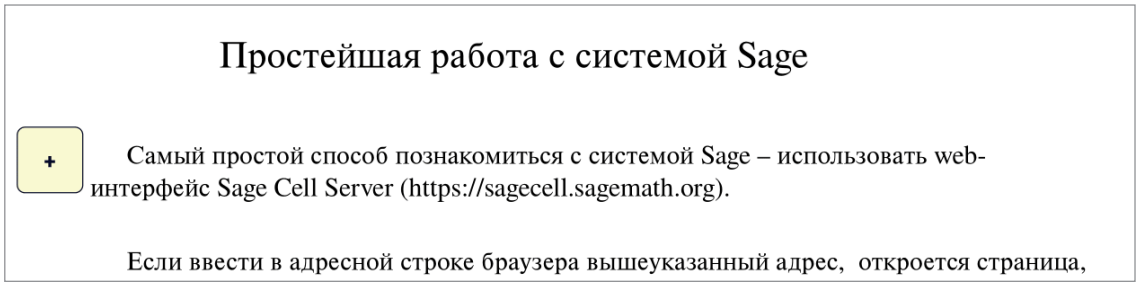


Рис. 1. Фрагмент эссе с нераскрытым расширением.

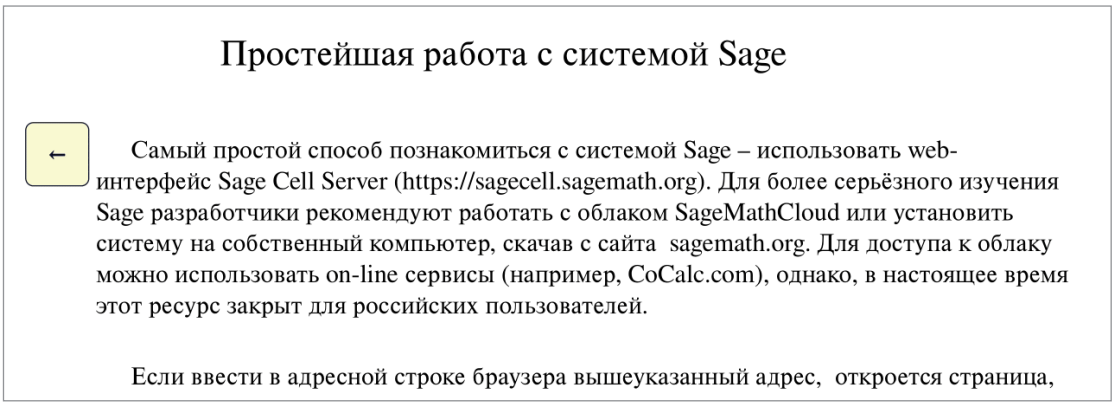


Рис. 2. Фрагмент эссе с раскрытым расширением.

Технически сложно включить в эссе рисунки. Наконец, трудно вводить текст эссе в XML-файл.

Во время работы над текстом, как правило, автор делает поправки, меняет слова, переформулирует фразы, переставляет местами части текста. При работе с расширяемым эссе этот процесс представляет собой большую сложность, так как вид текста в формате XML далек от текста на бумаге или в текстовом редакторе из-за обилия тегов. Найти в тексте нужное место довольно сложно; внесение любых изменений чревато появлением ошибок в структуре XML-файла.

Программное средство — редактор, который позволил бы работать с текстом расширяемого эссе, не отвлекаясь на моменты его технической реализации, был бы полезен потенциальным авторам. Ниже обсуждается программа-редактор 3E (extendable essay editor), предназначенная для создания расширяемых эссе, и расширяемое эссе о системе компьютерной алгебры Sage, созданное с помощью этого редактора.

4. ФУНКЦИОНАЛЬНОСТЬ РЕДАКТОРА 3E

Редактор предназначен для автора расширяемого эссе и должен обеспечить удобство его работы на протяжении всего периода создания расширяемого эссе.

Прежде всего, нужно скрыть технические подробности реализации расширяемого эссе, чтобы дать возможность автору сосредоточиться на проработке смыслового содержания самого эссе.

Редактор должен обеспечить удобный (традиционный) ввод и форматирование текста.

В редакторе должны быть предусмотрены средства для вставки в эссе формул и иллюстраций (рисунков, таблиц, графиков). Важно интегрировать иллюстративный материал внутрь HTML-файла, содержащего эссе. Читателю удобнее работать с одним файлом-эссе, чем следить за целостностью папки, содержащей файлы расширяемого эссе.

Написание расширяемого эссе – процесс длительный. Редактор должен давать возможность автору сохранить текущий вариант эссе с тем, чтобы продолжить работу с оставленного состояния.

Внешний вид эссе, с которым работает автор, должен отличаться от вида эссе, предъявляемого читателю (хотя бы потому, что автор должен видеть все расширения одновременно). Соответственно, редактор должен давать возможность автору работать с эссе в двух режимах: редактирование (набор текста, вставка расширений и т.п.); просмотр в том виде, как будет видеть эссе читатель, для проверки связности текста при раскрытии расширений.

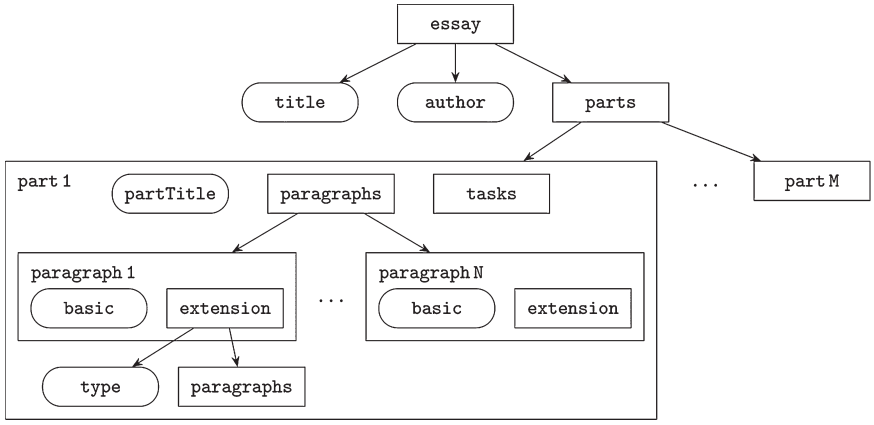


Рис. 3. Схема объекта “Расширяемое эссе” в редакторе.

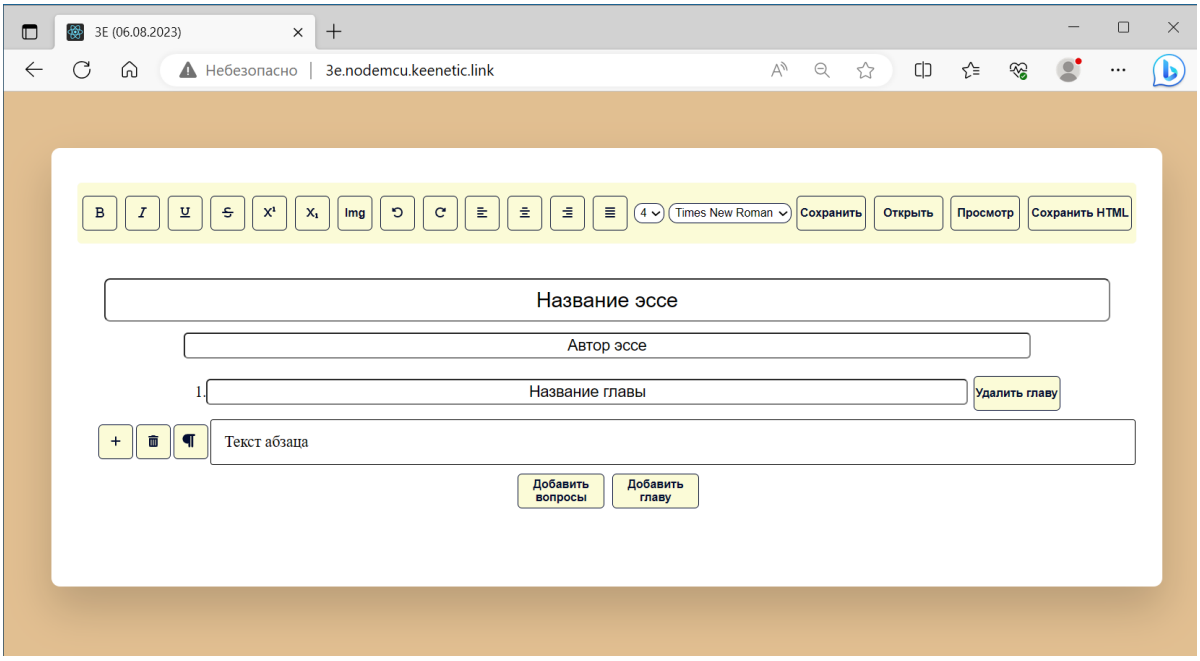


Рис. 4. Стартовая страница редактора 3E.

Расширяемое эссе предоставляется читателю в виде HTML-файла, следовательно, редактор должен генерировать и экспортировать внешнее представление расширяемого эссе в виде HTML-файла.

5. РЕАЛИЗАЦИЯ РЕДАКТОРА РАСШИРЯЕМЫХ ЭССЕ

Поскольку просмотр расширяемого эссе осуществляется в интернет-браузере, представляется логичным и редактирование расширяемых эссе реализовать в виде приложения для работы в браузере. У этого подхода есть как положительные, так и отрицательные стороны. Размещение редактора, выполненного в виде веб-приложения, на сервере дает возможность работать с ним на любом компьютере, подключенном к интернету; установка редактора на компьютер не требуется. Автору-пользователю не нужно обладать никакими специфическими навыками, достаточно просто открыть нужную страницу в браузере. С другой стороны, если отсутствует доступ к интернету, использовать редактор не удастся.

Учитывая все достоинства использования веб-технологий, было принято решение вести разработку редактора 3E в виде веб-приложения на языке JavaScript. В качестве основной библиотеки была выбрана популярная библиотека React, обеспечивающая эффективную обработку и отображение изменяющихся веб-страниц ([7]). React — библиотека с открытым исходным кодом, дающая возможность быстро и просто создать высокопроизводительные веб-приложения.

Расширяемое эссе представляется в виде JavaScript-объекта следующей структуры (рис. 3):

на самом верхнем уровне располагаются поля **title** и **author**, хранящие название эссе и автора. Содержание самого эссе представляется в виде массива разделов **parts**, каждый раздел является JavaScript-объектом со следующими полями: **partTitle** — название раздела, **paragraphs** — массив абзацев и **tasks** — контрольные вопросы. Каждый абзац в свою очередь также представляется объектом со следующей структурой: **basic** — базовое содержание абзаца в формате HTML, **extension** — расширение к абзацу. Если для абзаца расширения нет, то поле **extension** имеет значение **null**, в противном случае, **extension** представляет собой JavaScript-объект с атрибутами **type** (тип расширения) и **paragraphs** — массив абзацев расширения.

Поле **tasks** представляет собой массив JavaScript-объектов, каждый из которых содержит текст вопроса (поле **question**), массив вариантов ответов на вопрос (**answers**) с указанием правильного варианта (**rightAnswer**), а также пояснение к правильному ответу (**option**).

Поскольку для внутреннего представления эссе используется JavaScript-объект, в качестве формата хранения (внешнего представления) расширяемого эссе выбран JSON (JavaScript Object Notation) — текстовый формат для обмена данными, широко используемый в веб-приложениях.

Программа-редактор представляет собой несколько компонентов, из которых можно выделить следующие. Компонент **Options** — инструментальная панель редактора — осуществляет форматирование вводимого текста, а также сохранение/загрузку JSON-файла эссе. Компонент **ViewEssay** отображает эссе в режиме чтения. Компонент **EssayEditor** — основной компонент, сам редактор, отвечающий за отображение инструментальной панели и эссе в режиме редактирования, в котором осуществляется набор текста, добавление и удаление абзацев, добавление и удаление расширений всех типов, включая контрольные вопросы, а также добавление и удаление разделов эссе.

Для отображения формул используется библиотека KaTeX. Работа с формулами реализована так. В режиме редактирования эссе формулы вводятся в формате TEX. Формат TEX привычен для авторов, пишущих тексты на естественно-научные темы, содержащие формулы, т. к. часто издательства принимают статьи именно в этом формате. При отображении эссе в режиме просмотра библиотека KaTeX “на лету” транслирует формулы в формат MathML, позволяющий интернет-браузерам отображать формулы в привычном для человека виде.

6. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

При создании интерфейса редактора 3E авторы придерживались принципов, изложенных в [8]: узнаваемость, наглядность, лаконичность, традиционность. Использована светлая цветовая палитра для уменьшения зрительной нагрузки.

При запуске редактора 3E открывается окно, изображенное на рис. 4.

В верхней части редактора находится панель инструментов с кнопками управления. Ниже расположены поля для ввода названия эссе, указания автора (авторов) и, собственно, поля ввода текста эссе.

Кнопки панели инструментов сгруппированы в соответствии с их функциями: управление начертанием символов, выравнивание текста, работа с файлами.

6.1. Группа файловых команд

Редактор позволяет сохранить на компьютер автора текущее состояние эссе, открыть эссе для продолжения работы с ним, конвертировать эссе в формат HTML (рис. 5).

Кнопки “Сохранить” и “Открыть” имеют традиционный смысл. Внешний вид эссе в процессе редактирования несколько отличается от вида, предназначенного для чтения, поэтому в редакторе имеется возможность предварительного просмотра расширяемого эссе. При нажатии на кнопку “Просмотр”: в браузере открывается новая вкладка, в которой отображается то, как будет выглядеть эссе для читателя (рис. 6).

При этом в эссе реализована функциональность — можно открывать и закрывать расширения, отвечать на тестовые задания и т. п.

Кнопка “Сохранить HTML” вызывает генерацию HTML-файла, содержащего эссе; сгенерированный файл можно загрузить на компьютер пользователя (автора эссе) для последующей передачи читателям.

6.2. Ввод и редактирование текста эссе

При запуске редактора автоматически создается один раздел: поле названия раздела и поле для ввода текста первого абзаца. Закончив ввод текста раздела, можно добавить новый, нажав кнопку “Добавить раздел”, расположенную под текстом раздела, или создать тренировочные задачи (контрольные

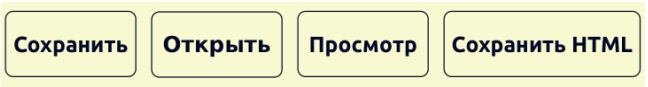


Рис. 5. Кнопки группы файловых команд.

вопросы), нажав кнопку “Добавить вопросы”. В редакторе предусмотрена возможность удаления раздела, причем раздел удаляется целиком, включая все абзацы и задачи, без каких-либо предупреждений. Для того чтобы исключить возможность ошибочного удаления раздела, кнопка “Удалить раздел” отличается от кнопок удаления абзацев по начертанию и размещена не слева на экране, а справа — правее названия раздела.

При создании раздела автоматически создается поле для ввода первого абзаца. При заполнении первой строки абзаца очередное слово переносится на новую строку, поле ввода увеличивается по вертикали, ввод продолжается в следующей строке. Работая с текстом абзаца, можно выделять, вставлять и удалять фрагменты текста, используя традиционные комбинации клавиш на клавиатуре компьютера. Выделенные фрагменты можно форматировать при помощи кнопок левой части панели инструментов вверху экрана (рис. 7).

Слева от поля ввода абзаца находятся кнопки “+”, “корзина” и “¶”, позволяющие создать расширение для текущего абзаца, удалить абзац (вместе с расширением) и добавить новый абзац.

6.3. Рисунки

Кнопка “Img” панели инструментов позволяет вставлять рисунки в текст эссе. Удобнее всего поместить рисунок в отдельный абзац. Рисунок можно расположить в центре страницы либо прижать к левому или правому краю. Также рисунки можно вставлять из буфера обмена с помощью сочетания клавиш Ctrl+V.

6.4. Формулы

Математические формулы (например, $\sin^2(x)$) набираются в формате TEX внутри знаков $...$: $\sin^2(x)$. Математический вид формулы можно увидеть в режиме просмотра эссе.

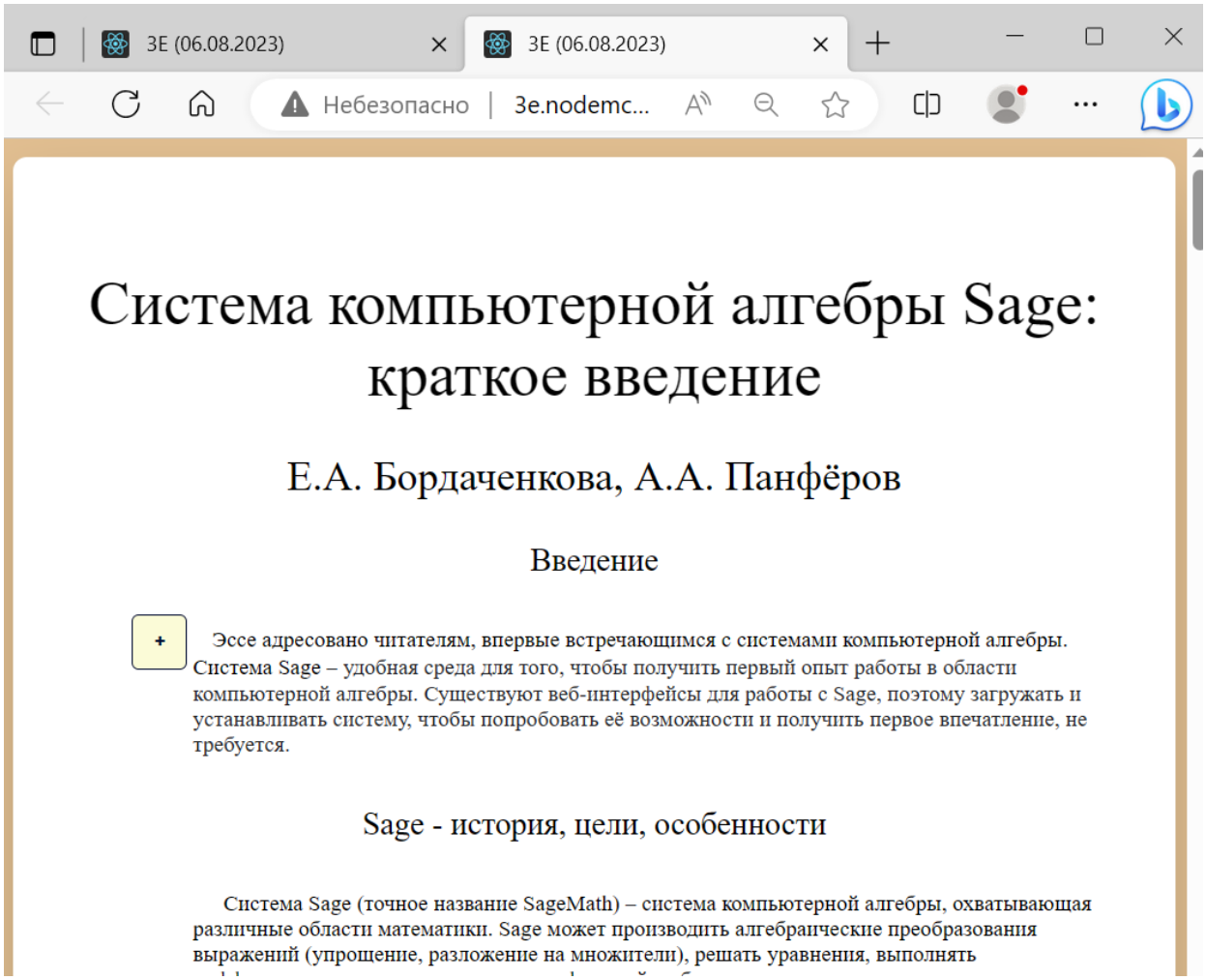


Рис. 6. Отображение расширяемого эссе в режиме просмотра.

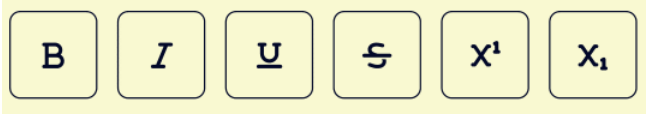


Рис. 7. Группа кнопок управления начертанием символов.

Ссылки на редактор расширяемых эссе 3E и на инструкцию к нему находятся на странице http://www.ccas.ru/ca/essay. Редактор запускается в любом современном браузере.

7. ЭССЕ “СИСТЕМА КОМПЬЮТЕРНОЙ АЛГЕБРЫ SAGE: КРАТКОЕ ВВЕДЕНИЕ”

7.1. Содержание эссе

С использованием редактора 3E было создано расширяемое эссе, посвященное основам работы с системой компьютерной алгебры Sage. При написании эссе мы опирались на книги В. Штейна [9], П. Циммермана и др. [10], А. Зобнина и др. [11] и использовали документацию сайта системы Sage [12]. В эссе описаны элементарные средства Sage и показана работа с многочленами и с матрицами.

Эссе адресовано читателям, впервые встречающимся с системами компьютерной алгебры. Система Sage — удобная среда для того, чтобы получить первый опыт работы в области компьютерной алгебры. Существуют веб-интерфейсы для работы с Sage, поэтому загружать и устанавливать систему, чтобы попробовать ее возможности и получить первое впечатление, не требуется.

Система Sage (точное название SageMath) — система компьютерной алгебры, охватывающая различные области математики, которая является свободно распространяемым открытым программным обеспечением. В создании и развитии системы принимают участие сотни ученых-разработчиков разных стран, проектом руководит Вильям Штейн.

Одна из идей, лежащих в основе Sage, состоит в том, чтобы максимально использовать существующие математические пакеты, вместо программирования всех алгоритмов с нуля. В Sage интегрировано более сотни специализированных и универсальных пакетов (например, GAP, PARI/GP, Singular, Maxima), что обеспечивает ее функциональную мощь. В качестве языка для работы с системой выбран Питон (Python). Использование универсального популярного языка программирования, поддержанного большим количеством библиотек, еще одна концепция, лежащая в основе Sage. Питон обладает большими выразительными возможностями; в эссе описаны конструкции, образующие логичное, на наш взгляд, подмножество, достаточное для начала освоения Sage.

Работа в системе Sage происходит в режиме диалога: пользователь вводит команду, система дает ответ. Команда — это приказ, например, вычислить значение выражения, построить матрицу или нарисовать график функции. Sage умеет работать с числовыми и с символьными выражениями, в эссе приведены соответствующие операции, проиллюстрированные примерами.

В разделе, посвященном многочленам, показаны операции, применимые к многочленам (перемножение многочленов, возведение в степень, вычисление наибольшего общего делителя двух многочленов и др.) Последний раздел эссе посвящен возможностям работы с матрицами — описание матриц, основные операции над ними, решение матричных уравнений.

Эссе “Система компьютерной алгебры Sage: краткое введение” доступно по адресу http://www.ccas.ru/ca/essay.

7.2. Использование редактора 3E

В процессе подготовки эссе редактор 3E запускался в браузерах Microsoft Edge (версии 115.0.1901.200, официальная сборка, 64-разрядная версия) и Firefox Browser (версии 116.0.2, 64-разрядная версия) под управлением операционных систем Windows 10 и Ubuntu 22.04.3 LTS соответственно.

При написании эссе “Система компьютерной алгебры Sage...”, кроме задачи создания самого эссе, стояла задача проверить работоспособность редактора 3E, в частности, работу с рисунками, формулами, расширениями и др. В эссе, помимо текстовых фрагментов, используются рисунки. Каждый рисунок оформлен в виде отдельного абзаца. Таким образом можно управлять размещением рисунка на странице с помощью команд форматирования абзацев. В эссе использованы два вида шрифтов — с засечками для ввода обычного текста и моноширинный для ввода программного кода. Ввод формул осуществлялся в формате TEX внутри символов $...$, в режиме просмотра формулы отображаются в привычном математическом виде. Максимальная глубина расширений — 1.

Затруднения возникали при вводе специфических символов, которые отсутствуют на клавиатуре (тех, которые вводятся в редакторе Word с помощью меню “Вставка символа”). Эти символы мы копировали из других текстовых редакторов.

В целом, редактор 3E показал свою работоспособность. В качестве направления дальнейшего развития инструмента видится отлаживание взаимодействия редактора с библиотекой KaTeX, возможный отказ от ее использования путем реализации трансляции формул в формат MathML, реализация возможности ввода символов наподобие того, как это устроено в редакторе Word, а также исправление замеченных ошибок.

БЛАГОДАРНОСТИ

Авторы выражают признательность А. А. Рябенко и Д. Е. Хмельнову за многократное плодотворное обсуждение работы. Благодарим рецензента за полезные советы. Наша особая благодарность — С. А. Абрамову, автору идеи расширяемого эссе, без которого работа над редактором 3E и над эссе “Система Sage...” не состоялась бы.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Проект реализуется по результатам грантового конкурса для преподавателей магистратуры 2022/2023 Стипендиальной программы Владимира Потанина.

СПИСОК ЛИТЕРАТУРЫ

1. *Войскунский А.Е., Солодов М.Ю.* Влияние свойств электронного текста на эффективность и результативность чтения: литературный обзор // Психология человека в образовании. 2020. Т. 2. № 2. С. 134–142. DOI: 10.33910/2686-9527-2020-2-2-134-142
2. *Miall D.S., Dobson T.* Reading hypertext and the experience of literature // Journal of Digital Information, 2001. V. 2. № 1. https://journals.tdl.org/jodi/index.php/jodi/article/view/35/37 (accessed 17.07.2023).
3. *Абрамов С.А., Бордаченкова Е.А., Хмельнов Д.Е.* Расширяемое эссе как гипертекстовая схема информационного и учебного материала // Журнал вычислительной математики и математической физики. 2013. Т. 53. № 3. С. 495–501.
4. *Абрамов С.А.* Расширяемое эссе “Сложность алгоритмов”, интернет-ресурс http://www.ccas.ru/sabramov/essay/, дата обращения 28.07.2023.
5. *Хмельнов Д.Е.* Расширяемое эссе “Реализация расширяемого эссе”, интернет-ресурс http://www.ccas.ru/ca/_media/essayonessay_html.rar, дата обращения 28.07.2023.
6. *Зубарева В.Н.* Расширяемое эссе “Портрет Адели Блох-Бауэр I”, интернет-ресурс http://www.ccas.ru/ca/_media/portrait.rar, дата обращения 28.07.2023.
7. *Бенкс А., Порсело Е.* React и Redux: функциональная веб-разработка, СПб.: Питер, 2018. 336 с.
8. *Джонсон Д.* Умный дизайн: Простые приемы разработки пользовательских интерфейсов, СПб.: Питер, 2012. 224 с.
9. *Stein W.* Sage for Power Users, 2012, https://wstein.org/books/sagebook/sagebook.pdf, дата обращения 04.08.2023.
10. *Zimmermann P, Casamayou A., Cohen N., Connan G., Dumont T., Fousse L., Maltey F., Meulien M., Mezzarobba M., Pernet C., Thiéry N., Bray E., Cremona J., Forets M., Ghitza A., Thomas H.* Computational mathematics with SageMath. Society for Industrial and Applied Mathematics, 2018, https://www.sagemath.org/sagebook/english.html, дата обращения 04.08.2023.
11. *Голубков А.Ю., Зобнин А.И., Соколова О.В.* Компьютерная алгебра в системе Sage, М.: МГТУ им. Н.Э.Баумана, 2013. 80 стр.
12. Интернет-ресурс SageMath Feature Tour, https://www.sagemath.org/tour.html, дата обращения 04.08.2023.

AN EXTENDABLE ESSAY ON THE SAGE COMPUTER ALGEBRA SYSTEM AND AN EDITOR FOR CREATING EXTENDABLE ESSAYS

© 2024 E. A. Bordachenkova[a], V. N. Zubareva[a], A. A. Panferov[a]

[a]*Department of Computational Mathematics and Cybernetics, Moscow State University, Moscow, 119991 Russia*

An extendable essay is a special format of electronic texts that is more convenient for reading than hypertext. To facilitate the creation and editing of extendable essays, an editor program implemented as a web application is proposed. Using this editor, an extendable essay on the Sage computer algebra system is written. Sage seems to be a good choice for the users who are not familiar with computer algebra systems.

Keywords: extendable essay, electronic text, essay editor, Sage

REFERENCES

1. *Voiskunskii A.E., Solodov M.Yu.* Influence of electronic text properties on reading efficiency and effectiveness: A literature review, Psikhologiya Chel. Obraz. 2020. V. 2. № 2. P. 134–142. https://doi.org/10.33910/2686-9527-2020-2-2-134-142

2. *Miall D.S., Dobson T.* Reading hypertext and the experience of literature // J. Digital Inf. 2001. V. 2. № 1. https://journals.tdl.org/jodi/index.php/jodi/article/view/35/37.
3. *Abramov S.A., Bordachenkova E.A., Khmel'nov D.E.* Extendable essay as a hypertext scheme for information and educational material // Comput. Math. Math. Phys. 2013. V. 53. P. 369–374.
4. *Abramov S.A.* Extendable essay "Complexity of algorithms." http://www.ccas.ru/sabramov/essay.
5. *Khmelэnov D.E.* Extendable essay "Implementation of an extendable essay." http://www.ccas.ru/ca/_media/essayonessay_html.rar.
6. *Zubareva V.N.* Extendable essay "Portrait of Adele Bloch-Bauer I." http://www.ccas.ru/ca/_media/portrait.rar.
7. *Banks A., Porcello E.* Learning React: Functional Web Development With React and Redux, Oreilly & Associates, 2017.
8. *Johnson J.* Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules, Amsterdam: Elsevier, 2010.
9. *Stein W.* Sage for power users, 2012. https://wstein.org/books/sagebook/sagebook.pdf.
10. *Zimmermann P., Casamayou A., Cohen N., Connan G., Dumont T., Fousse L., Maltey F., Meulien M., Mezzarobba M., Pernet C., Thiéry N., Bray E., Cremona J., Forets M., Ghitza A., Thoma H.* Computational mathematics with SageMath, Society for Industrial and Applied Mathematics, 2018. https://www.sagemath.org/sagebook/english.html.
11. *Golubkov A.Yu., Zobnin A.I., Sokolova O.V.* Komp'yuternaya algebra v sisteme Sage (Computer Algebra in the Sage System), Moscow: MGTU N. E. Baumana, 2013.
12. SageMath feature tour. https://www.sagemath.org/tour.html.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 22–32*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

ИНТЕГРИРОВАНИЕ ВЫРОЖДЕННОЙ СИСТЕМЫ ОДУ

© 2024 г. А. Д. Брюно[a], *, В. Ф. Еднерал[b], **

[a] *Институт прикладной математики им. М.В. Келдыша РАН*
125047 Москва, Миусская площадь, д. 4

[b] *Научно-исследовательский институт ядерной физики им. Д.В. Скобельцына*
Московского государственного университета им. М. В. Ломоносова
119991 Москва, ГСП-1, Ленинские горы, д. 1(2)

**E-mail: abruno@keldysh.ru*
***E-mail: edneral@theory.sinp.msu.ru*

Поступила в редакцию 24.08.2023
После доработки: 10.09.2023
Принята к публикации 01.10.2023

Изучается интегрируемость автономной двумерной полиномиальной системы ОДУ с вырожденной особой точкой в начале координат, зависящей от шести параметров. Условие интегрируемости первого квазиоднородного приближения позволяет фиксировать один из параметров на счетном множестве его значений. Дальнейший анализ проводится для одного такого значения и пяти свободных параметров. С помощью метода степенной геометрии система приводится к невырожденной форме при помощи процесса расщепления (blowup). Далее методом нормальной формы вычисляются необходимые условия локальной интегрируемости. Иными словами, ищутся такие условия на параметры, при которых исходная система является локально интегрируемой вблизи вырожденной стационарной точки. В результате решения этих условий мы нашли семь двухпараметрических семейств в пятимерном параметрическом пространстве. При значениях параметров из этих семейств были найдены первые интегралы системы. Громоздкие вычисления, возникающие в обсуждаемой задаче, были выполнены средствами компьютерной алгебры.

Ключевые слова: вырожденная система ОДУ, методы нормальной формы и степенной геометрии
DOI: 10.31857/S0132347424020044 **EDN:** RPDZLZ

1. ВВЕДЕНИЕ

При исследовании нелинейных систем ОДУ желательно получить точные решения таких систем в виде формул. Если такие решения найдены, можно изучать свойства системы вблизи этих решений. К сожалению, для большинства систем ОДУ в общем виде точные решения найти не удается. Но когда система зависит от параметров, мы можем попытаться найти семейства значений параметров, при которых такая система допускает точное решение. Здесь мы предлагаем и проверяем новый подход к поиску таких семейств. Отметим, что если под точным решением подразумевать существование решения в квадратурах, то для автономной системы ОДУ вопрос сводится к изучению ее интегрируемости. Но для интегрируемости двумерной автономной динамической системы достаточно иметь один первый интеграл движения, поэтому задача о точных решениях сводится здесь к поиску случаев интегрируемости и вычислению соответствующих интегралов.

В работах Пуанкаре и более поздних математиков была сформулирована идея приведения системы ОДУ к нормальной форме – к виду, наиболее простому для исследования. Приведенная к нормальной форме система, по сути, имеет порядок на единицу меньше исходной. В случае разрешенной относительно производных автономной двумерной системы она преобразуется к автономной одномерной и становится, таким образом, интегрируемой в квадратурах.

Нормализующее преобразование, как правило, записывается в виде бесконечного формального ряда и возникает вопрос о его сходимости. Условия сходимости нормализующего преобразования получены и исследованы в работах [1, 2] и выполняются далеко не всегда. Однако если система ОДУ зависит от параметров, существует возможность, что при некоторых значениях параметров эти условия все же выполняются. Иными словами, мы можем записать условия интегрируемости в виде условия на параметры системы. Все сказанное относится к локальному анализу, в котором изучение системы ОДУ производится в малых окрестностях некоторых точек фазового пространства, причем наибольший интерес представляют собой окрестности неподвижных точек, в которых существуют решения, не

зависящие от независимой переменной, то есть решения – константы. Итак, возможно записать условие локальной интегрируемости как некоторое алгебраическое условие на параметры системы. Но если система имеет несколько неподвижных точек, то нужно искать семейства параметров, при которых система локально интегрируема во всех этих точках одновременно. Если такие семейства существуют, то система, возможно, глобально интегрируема при соответствующих значениях параметров. Интегралы системы при этих значениях параметров следует затем искать другими методами. Эту схему удается реализовать на практике.

Для иллюстрации указанного подхода мы рассматриваем вырожденную двумерную систему автономных обыкновенных дифференциальных уравнений, разрешенную относительно производных и с полиномиальной правой частью пятого порядка. Система зависит от шести параметров. Один из параметров удается фиксировать исходя из условия интегрируемости первого квазиоднородного приближения. Пять параметров затем рассматриваются свободными.

Выбор такой системы определялся, во-первых, тем, что она принадлежит к классу ОДУ без линейной части. Такой класс систем ранее практически не исследовался. Во-вторых, после расщепления эта система распадается на несколько нильпотентных систем, которые мы и исследуем. Таким образом мы проверяем подход также и для нильпотентных систем.

Настоящая работа завершает цикл статей [3, 4, 5]. В ней закончено рассмотрение всех полученных случаев интегрируемости рассматриваемой ниже вырожденной системы ОДУ. Исследование последнего из них потребовало изучения резонанса 27-го порядка, формула нормализующего преобразования для которого содержит свыше 8 миллионов слагаемых. Эта работа заняла более года. Результат (первый интеграл) оказался, впрочем, вполне компактным, см. формулу (9.14). Очевидно, что столь громоздкие символьные вычисления невозможны без применения систем компьютерной алгебры. Безальтернативность использования компьютерных систем символьных вычислений подчеркивалась также в работе [6].

Применение метода нормальной формы к исследованию интегрируемости систем нелинейных ОДУ рассматривалось в работе [7]. Обсуждаемый подход был также успешно применен к поиску интегрируемых случаев обобщенного уравнения Льенара [8, 9, 10]. Исследована также система Баутина [11]. Примером доказательства неинтегрируемости системы ОДУ методом номальной формы может служить работа [12].

2. ИССЛЕДУЕМАЯ СИСТЕМА

Мы будем рассматривать систему

$$\begin{aligned} \frac{dx}{dt} &= \alpha y^3 + \beta x^3 y + a_0 x^5 + a_1 x^2 y^2, \\ \frac{dy}{dt} &= \gamma x^2 y^2 + \delta x^5 + b_0 x^4 y + b_1 x y^3. \end{aligned} \tag{2.1}$$

Эта система является подсемейством системы, представленной в работе [13]. x и y здесь зависимые переменные, функции независимой переменной – времени t. Системы с нильпотентной матрицей линейной части подробно исследованы Ляпуновым и другими авторами. Система (2.1) представляет собой простейший случай двумерной системы без линейной части с ломаной Ньютона (см. главу II в [2]), состоящей из единственного ребра. В общем случае такие системы не изучались. Однако система с подобным носителем рассматривалась в [13], где авторы положили $-\alpha = \beta = 1$ и $3\beta + 2\gamma = 0$ и изучили гамильтонов подслучай этой системы с дополнительным предположением, что соответствующий гамильтониан свободен от квадратичных множителей. Мы изучаем здесь общий случай, когда система (2.1) негамильтонова.

3. УПРОЩЕНИЕ ЗАДАЧИ

Рассмотрим первое квазиоднородное приближение системы (2.1) и необходимые условия его локальной интегрируемости. Это приближение имеет вид [3]

$$\frac{d\tilde{x}}{dt} = \alpha \tilde{y}^3 + \beta \tilde{x}^3 \tilde{y}, \quad \frac{d\tilde{y}}{dt} = \gamma \tilde{x}^2 \tilde{y}^2 + \delta \tilde{x}^5, \tag{3.1}$$

где $\alpha \neq 0$ и $\delta \neq 0$. Используя линейное преобразование $x = \sigma \tilde{x}$, $y = s\tilde{y}$, мы можем исключить эти два ненулевых параметра и вместо (3.1) получить систему

$$\frac{dx}{dt} = -y^3 - bx^3 y, \quad \frac{dy}{dt} = cx^2 y^2 + x^5. \tag{3.2}$$

Любая двумерная автономная квазиоднородная система имеет первый интеграл, но не обязательно гладкий. Нас интересует достаточно гладкая интегрируемость системы (3.1), во всяком случае без существенных особенностей в начале координат, поскольку мы ищем условия на параметры, при которых система (2.1) является гладко интегрируемой.

В работе [3] был доказан следующий результат.

Теорема 1. *В случае* $D(3b+2c)^2-24\neq 0$ *система* (3.2) *аналитически интегрируема тогда и только тогда, когда число* $N=(3b-2c)/\sqrt{D}$ *рационально.*

В этой статье мы изучим частный случай $c=1/b$, $b\neq 0$, когда $N=1$ и $D=(3b-2/b)^2$. В силу теоремы 1 первое квазиоднородное приближение имеет аналитический интеграл, хотя это не гамильтонова система. Случай $b^2=2/3$, при котором $D=0$, будет рассмотрен отдельно.

Далее мы будем изучать проблему интегрируемости упрощенной системы, в которой остается 5 свободных параметров

$$\begin{aligned}\frac{dx}{dt}&=-y^3-bx^3y+a_0x^5+a_1x^2y^2,\\ \frac{dy}{dt}&=(1/b)x^2y^2+x^5+b_0x^4y+b_1xy^3.\end{aligned}\tag{3.3}$$

Таким образом, теперь мы рассматриваем систему с пятью произвольными параметрами a_i,b_i, $(i=0,1)$ и $b\neq 0$.

4. ПРИВЕДЕНИЕ СИСТЕМЫ К НЕВЫРОЖДЕННОМУ ВИДУ

Это можно сделать, используя расщепление (см. Главу 1, п. 1.8 в [2]) при помощи обратимого степенного преобразования

$$x=uv^2,\quad y=uv^3\tag{4.1}$$

с масштабированием времени $u^2v^7dt=d\tau$. В результате система (3.3) преобразуется к виду

$$\begin{aligned}\frac{du}{d\tau}&=-3u-(3b+2/b)u^2-2u^3+\\ &+(3a_1-2b_1)u^2v+(3a_0-2b_0)u^3v,\\ \frac{dv}{d\tau}&=v+(b+1/b)uv+u^2v+\\ &+(b_1-a_1)uv^2+(b_0-a_0)u^2v^2.\end{aligned}\tag{4.2}$$

Следует изучить эту систему на инвариантных линиях $u=0$ и $v=0$.

Техника определения конкретного вида степенного преобразования для расщепления вырожденной точки описана в § 5 главы III книги [14].

5. О НОРМАЛЬНОЙ ФОРМЕ И УСЛОВИИ СХОДИМОСТИ НОРМАЛИЗУЮЩЕГО ПРЕОБРАЗОВАНИЯ

В окрестности неподвижной точки $X=0$ система вида (4.2) всегда может быть записана в обозначениях

$$\frac{dX}{dt}=AX+\Phi(X),\tag{5.1}$$

где $X=(x_1,\ldots,x_n)$ – вектор переменных, A – матрица линейных членов системы, а степенной ряд $\tilde{\Phi}(X)$ не содержит свободных и линейных по X членов. Линейной заменой переменных

$$X=BY,\tag{5.2}$$

приведем матрицу A к жордановой форме $J=B^{-1}AB$ и систему (5.1) к виду

$$\frac{dY}{dt}=JY+\tilde{\Phi}(Y).\tag{5.3}$$

Формальная квазитождественная замена координат

$$Y=Z+\Xi(Z),\tag{5.4}$$

где $\Xi=(\xi_1,\ldots,\xi_n)$ – вектор, а $\xi_j(Z)$ – формальные степенные ряды, приводит (5.3) к нормальной форме [1, 2]

$$\frac{dZ}{dt}=JZ+\Psi(Z).\tag{5.5}$$

Линейная часть системы при таком преобразовании не меняется.

Мы запишем эту нормальную форму в виде

$$\frac{dz_j}{dt}=z_jg_i(Z)=z_j\sum_{Q\in N_j}g_{jQ}Z^Q,\quad j=1,\ldots,n.\tag{5.6}$$

$$N_j=\left\{Q:Q\in\mathbb{Z}^n,\ Q+E_j\geq 0\right\},\ j=1,\ldots,n.$$

E_j означает единичный вектор с j-той ненулевой компонентой. Другими словами, $Q=(q_1,\ldots,q_n)$ – это вектор с целочисленными компонентами, которые все неотрицательны за исключением j-того элемента, он может быть равен -1. Мы используем мультииндексную запись $Z^Qz_1^{q_1}\ldots z_n^{q_n}$. В формуле (5.6) мы предположили, что J является диагональной матрицей. Это предположение сделано лишь для упрощения записи, поскольку в данной работе необходимости в рассмотрении жорданова случя не возникает. Имеются формулы и для общего жорданового случая.

Обозначим вектор, состоящий из собственных значений матрицы A и являющийся диагональю матрицы J, как $\Lambda=(\lambda_1,\ldots,\lambda_n)$.

Система (5.6) называется *резонансной нормальной формой*, если:

а) J — жорданова матрица,

б) в записи (5.6) есть только *резонансные члены* z_{ji} g_{jQ} Z^Q, для которых скалярное произведение векторов обнуляется

$$\langle Q,\Lambda\rangle=q_1\lambda_1+\ldots+q_n\lambda_n=0.\tag{5.7}$$

Теорема 2 ([1, 2]). *Существует формальное преобразование* (5.4)*, приводящее систему к ее нормальной форме* (5.6).

В работах [1, 2] сформулировано условие **A** на коэффициенты нормальной формы (5.6) и условие ω на собственные значения матрицы A. Выполнение этих двух условий гарантирует сходимость нормализующего преобразования (5.4).

Условие A. *В нормальной форме* (5.6)

$$g_j = \lambda_j \alpha(Z) + \bar{\lambda}_j \beta(Z), \ \ j = 1,\ldots,n, \qquad (5.8)$$

где $\alpha(Z)$ *и* $\beta(Z)$ *– некоторые степенные ряды, не зависящие от индекса* j.

Пусть

$$\omega_k = \min\left|\langle Q,\Lambda\rangle\right| \text{ по всем } Q \in N_1 \cup \ldots \cup N_n,$$
$$\langle Q,\Lambda\rangle \neq 0, \sum_{j=1}^{n} q_j < 2^k, k = 1,2,\ldots$$

Условие ω (условие на малые знаменатели). *Сходится ряд*

$$\sum_{k=1}^{\infty} 2^{-k} \log\omega_k > -\infty,$$

Это слабое арифметическое условие на собственные числа Λ. В двумерном резонансном случае условие ω выполняется, и далее здесь мы его обсуждать не будем.

Теорема 3 ([1, 2]). *Если у системы* (5.3) *вектор* Λ *удовлетворяет условию* ω *и нормальная форма* (5.5) *удовлетворяет условию* **A***, то нормализующее преобразование* (5.4) *аналитично.*

Рассмотрим подробнее условие **A** в двумерном случае. Пусть отношение соответствующих двух собственных чисел рационально

$$\lambda_1/\lambda_2 = -m/n \quad \text{или} \quad n\lambda_1 = -m\lambda_2, \qquad (5.9)$$

где m и n – натуральные числа. Тогда, учитывая (5.6) и (5.9), мы можем написать условие **A** (5.8) в виде системы

$$\begin{aligned} n\frac{d\log(z_1)}{dt} &= n\left(\lambda_1 \cdot \alpha(Z) + \bar{\lambda}_1 \cdot \beta(Z)\right), \\ m\frac{d\log(z_2)}{dt} &= m\left(\lambda_2 \cdot \alpha(Z) + \bar{\lambda}_2 \cdot \beta(Z)\right). \end{aligned} \qquad (5.10)$$

Суммируя эти два равенства, вследствии (5.9), получим

$$\frac{d\log(z_1^n z_2^m)}{dt} = 0 \quad \text{или} \quad z_1^n z_2^m = \text{const}. \qquad (5.11)$$

Итак, в нормализованной системе $z_1^n z_2^m$ – первый интеграл движения. Это соответствует локальной интегрируемости исходной системы в начале координат, поскольку условия **A** достаточно для сходимости нормализующего преобразования вблизи неподвижной точки. В случае же нулевого собственного значения $\lambda_i = 0$, условие **A** имеет вид $g_i(Z) = 0$ и тогда в нормализованной системе z_i – первый интеграл движения.

6. УСЛОВИЕ ИНТЕГРИРУЕМОСТИ

Условие **A** является необходимым и достаточным условием локальной интегрируемости двумерной системы вблизи элементарной неподвижной точки [1, 2]. Это условие является сильным алгебраическим условием на коэффициенты нормальной формы. Для локальной интегрируемости исходной системы (3.3) вблизи вырожденной (неэлементарной) стационарной точки необходимо иметь локальную интегрируемость вблизи каждой из элементарных стационарных точек, которые возникают в результате описанного выше процесса расщепления.

Условие **A** представляет собой бесконечный ряд полиномиальных уравнений на парметры системы. Совокупность соответствующих многочленов образует идеал над полиномиальным кольцом. В силу теоремы Гильберта о базисе [16], такая бесконечная система эквивалентна конечной. Проблема состоит в том, что мы можем лишь предполагать, что многочлены низлежащих порядков такой системы уже образуют ее базис. Однако изученные нами примеры двумерных систем ОДУ подтверждают такое предположение. В любом случае, если условие **A** в виде бесконечной совокупности уравнений является необходимым и достаточным, то любое конечное подмножество соответствующих уравнений будет необходимым условием. Это будет конечная система алгебраических уравнений относительно параметров исходной системы, ее решения будут соответствовать необходимым условиям локальной интегрируемости системы вблизи соответствующей неподвижной точки расщепленной системы (4.2), а следовательно – необходимым условиям интегрируемости исходной системы (3.3). Но вырожденной точке системы (3.3) соответствуют несколько неподвижных точек системы (4.2), следовательно искомым условием интегрируемости будет выполнение соответствующих условий во всех неподвижных точках одновременно.

7. УСЛОВИЕ ИНТЕГРИРУЕМОСТИ ДЛЯ РАСЩЕПЛЕННОЙ СИСТЕМЫ

При степенном преобразовании (4.1) неподвижная точка системы (3.3) $x = y = 0$ раздувается в две инвариантные прямые $u = 0$ и $v = 0$. Вдоль

прямой $u=0$ система (4.2) имеет единственную стационарную точку $u=v=0$. Вдоль второй линии $v=0$ эта система имеет четыре элементарных стационарных точки

$$u=0,\quad u=-\frac{1}{b},\quad u=-\frac{3b}{2},\quad u=\infty. \qquad (7.1)$$

Итак, необходимым условием локальной интегрируемости системы (3.3) вблизи точки $x=y=0$ является локальная интегрируемость вблизи этих неподвижных точек одновременно.

Лемма 1 ([3]). *Вблизи точек* $u=v=0$ *и* $u=\infty, v=0$ *система* (4.2) *локально интегрируема.*

Таким образом, мы должны исследовать условия локальной интегрируемости на двух других стационарных точках (7.1). Тогда у нас будут условия локальной интегрируемости системы (3.3) вблизи точки $X=0$.

Рассмотрим стационарную точку $u=-1/b$, $v=0$. Сначала мы ограничимся случаем $b^2=2/3$, когда линейная часть системы (4.2) после сдвига $u=w-1/b$ не имеет чисто нулевых собственных значений. При $b^2=2/3$ преобразованная система в новых переменных $\tilde{u}$ и v в линейной части имеет ячейку Жордана с обоими нулевыми собственными значениями. Этот случай мы изучим ниже с помощью еще одного степенного преобразования.

Чтобы упростить собственные значения, мы еще раз отмасштабируем время множителем $d\tau=(2-3b^2)/b^2\,d\tau_1$. После этого получим вектор собственных значений системы (4.2) в этой точке в виде $(\lambda_1,\lambda_2)=(-1,0)$. Таким образом, нормальная форма системы будет иметь вид

$$\begin{aligned}\frac{dz_1}{d\tau_1}&=-z_1+z_1 g_1(z_2),\\ \frac{dz_2}{d\tau_1}&=\quad z_2 g_2(z_2),\end{aligned} \qquad (7.2)$$

где $g_{1,2}(x)$ – формальные степенные ряды от x. Коэффициенты этих рядов являются рациональными функциями параметров системы a_0,a_1,b_0,b_1 и b. Можно доказать, что знаменатель каждой из этих рациональных функций пропорционален некоторой целой степени $k(n)$ многочлена $(2-3b^2)$. Их числители являются полиномами от параметров системы

$$g_{1,2}(x)=\sum_{n=1}^{\infty}\frac{p_{1,2;n}(b,a_0,a_1,b_0,b_1)}{\left(2-3b^2\right)^{k(n)}}x^n.$$

Условие **A** интегрируемости уравнения (7.2) выглядит как $g_2(x)\equiv 0$. Это эквивалентно бесконечной полиномиальной системе уравнений

$$p_{2,n}(b,a_0,a_1,b_0,b_1)=0,\quad n=1,2,\ldots. \qquad (7.3)$$

Мы вычислили первые три полинома $p_{2,1}, p_{2,2}, p_{2,3}$ с помощью нашей программы [15] и получили два решения соответствующего конечного подмножества уравнений (7.3) при $b\neq 0$

$$a_0=0,\quad a_1=-b_0 b,\quad b_1=0,\quad b^2\neq 2/3 \qquad (7.4)$$

и

$$a_0=a_1 b,\quad b_0=b_1 b,\quad b^2\neq 2/3. \qquad (7.5)$$

Добавление следующего уравнения $p_{2,4}=0$ к подмножеству уже решенных уравнений не меняет эти решения. Мы предполагаем, что полученные первые уравнения уже образуют тот конечный базис, который должен существовать у полиномиальной системы по теореме Гильберта [16].

Построение многочленов $p_{2,n}(b,a_0,a_1,b_0,b_1)$ общего вида для больших порядков n весьма сложная технически проблема. Значительно проще построить эти полиномы при условиях (7.4) и (7.5) для фиксированных значений параметра b. Мы проверили решения подмножества уравнений

$$p_{2,n}(b,a_0=a_1 b,a_1,b_0=b_1 b,b_1)=0,$$
$$n=1,\,2,\ldots,\,28,$$

при $b=1$ и $b=2$. Во всяком случае до 28-го порядка все уравнения выполняются, поэтому предположим, что в неподвижной точке $u=-1/b$, $v=0$ решения (7.4) и (7.5) удовлетворяют условию **A**.

Рассмотрим вторую неподвижную точку $u=-3b/2$, $v=0$. Изменим масштаб времени $d\tau=(2-3b^2)d\tau_2$. Получим вектор собственных значений системы (4.2) в этой точке в виде $(-1/4, 3/2)$. Таким образом нормальная форма имеет резонанс седьмого порядка.

$$\begin{aligned}\frac{dz_1}{d\tau_2}&=-(1/4)z_1+z_1 r_1\left(z_1^6 z_2\right),\\ \frac{dz_2}{d\tau_2}&=(3/2)z_2+z_2 r_2\left(z_1^6 z_2\right),\end{aligned} \qquad (7.6)$$

где $r_{1,2}(x)$ также формальные степенные ряды, причем в (7.6) они зависят от единственной резонансной переменной $z_1^6 z_2$.

$$r_{1,2}(x)=\sum_{n=1}^{\infty}\frac{q_{1,2;n}(b,a_0,a_1,b_0,b_1)}{(2-3b^2)^{l(n)}}x^n.$$

Условие **A** для уравнения (7.6) выглядит как $6r_1(x)+r_2(x)=0$. Оно эквивалентно бесконечной полиномиальной системе уравнений

$$6q_{1,n}(b,a_0,a_1,b_0,b_1)+q_{2,n}(b,a_0,a_1,b_0,b_1)=0,$$
$$n=7,14,\ldots. \tag{7.7}$$

Мы вычислили многочлены $q_{1,7}, q_{2,7}$ и решили соответствующие уравнения из множества (7.7) при значениях параметров из решения (7.5). Мы нашли пять различных двухпараметрических решений (b и a_1 – свободные параметры)

$$\begin{aligned}&1)\ b_1=-2a_1, a_0=a_1b, b_0=b_1b, b^2\neq 2/3,\\ &2)\ b_1=(3/2)a_1, a_0=a_1b, b_0=b_1b, b^2\neq 2/3,\\ &3)\ b_1=(8/3)a_1, a_0=a_1b, b_0=b_1b, b^2\neq 2/3\end{aligned} \tag{7.8}$$

и

$$\begin{aligned}&4)\ b_1=\frac{197-7\sqrt{745}}{24}a_1, a_0=a_1b, b_0=b_1b, b^2\neq 2/3,\\ &5)\ b_1=\frac{197+7\sqrt{745}}{24}a_1, a_0=a_1b, b_0=b_1b, b^2\neq 2/3.\end{aligned} \tag{7.9}$$

Мы проверили выполнение (7.7) до $n=49$ для решений (7.8) при значениях $b=1$ и $b=2$ и произвольном (символьном) a_1.

В то же время решения (7.8) являются частными случаями решения (7.5) при соответствующих значениях b_1, а решение (7.4) – частный случай (7.8) при значениях $a_0=b_1=0$, т.е. полученные решения справедливы для обеих точек одновременно.

Решения (7.9), начиная с порядка $n=14$, верны только при дополнительном условии $a_1=0$. Но при этом решения (7.9) – лишь частный случай решения (7.8). Итак, в соответствии с основным предположением мы можем предположить

Гипотеза 1. *При $b\neq 0$ и $b^2\neq 2/3$, выполнение какого-либо из равенств:*

$$\begin{aligned}&1)\ b_1=0,\quad a_0=0,\quad a_1=-b_0;\\ &2)\ b_1=-2a_1,\quad a_0=a_1b,\quad b_0=-2a_1b;\\ &3)\ b_1=(3/2)a_1,\quad a_0=a_1b,\quad b_0=(3/2)a_1b;\\ &4)\ b_1=(8/3)a_1,\quad a_0=a_1b,\quad b_0=(8/3)a_1b.\end{aligned} \tag{7.10}$$

является условием локальной интегрируемости системы (4.2) *во всех стационарных точках многообразия $v=0$ и, таким образом, является необходимым условием локальной интегрируемости системы* (3.3) *вблизи вырожденной точки $x=y=0$.*

Итак, каждое из условий (7.10) является следствием условия **A**, поэтому, чтобы доказать эту гипотезу, нам необходимо доказать что при выполнении любого из условий (7.10) выполняется условие **A**. Т.е. что при решении всей бесконечной цепочки уравнений ни одно из условий (7.10) не исчезнет, в то же время и новых решений у системы при ее пополнении появиться не может. В силу существования у бесконечной нетеровой системы конечного базиса [16], такое доказательство представляется возможным, но пока мы его не нашли. Обнаружилась, однако, другая возможность применения соотношений (7.10).

8. ДОСТАТОЧНЫЕ УСЛОВИЯ ГЛОБАЛЬНОЙ ИНТЕГРИРУЕМОСТИ

Заметим, что если согласиться с гипотезой 1, то можно сформулировать следующее утверждение

Гипотеза 2. *Условия* (7.10) *являются необходимыми условиями существования достаточно гладкого первого интеграла системы* (4.2).

Действительно, если предположить существование функции, однозначной в некоторой области и представляющей собой интеграл движения, то в каждой точке этой области, включая неподвижные, должна иметь место локальная интегрируемость. Первые интегралы для каждого случая из (7.10) выписаны в Приложении. Интегралы были вычислены методами Лагутинского [17] или Дарбу [4, 18] при помощи системы MATHEMATICA-11.

Заменой переменных, обратной к (4.1), мы получили также первые интегралы исходной вырожденной системы (3.3). Заметим, что для каждого такого случая из выражения для первого интеграла можно выписать решение системы в квадратурах, т.е. мы проинтегрировали исходную систему для некоторых двумерных множеств параметров. Еще три первых интеграла получены ниже для случая $b^2=2/3$.

9. СЛУЧАЙ $b^2=2/3$

Заметим, что выбор знака b не имеет значения из-за линейного автоморфизма $\{x,y,b,a_0,a_1,b_0,b_1\}\to$ $\to\{-x,-y,-b,a_0,-a_1,b_0,-b_1\}$ системы (3.3). Положим ниже $b=+\sqrt{2/3}$. В этом случае обе стационарные точки $u=-3b/2$, $v=0$ и $u=-1/b$, $v=0$ схлопываются в одну и после сдвига $u\to w-1/b$ вместо (4.2) имеем систему

$$\begin{aligned}\frac{dw}{d\tau}=&v\left(-92\sqrt{32}a_0+92a_1+3\sqrt{32}b_0-3b_1\right)+\\ &+wv\left(272a_0-3\sqrt{6}a_1-9b_0+2\sqrt{6}b_1\right)+\\ &+w^2v\left(-9\sqrt{32}a_0+3a_1+3\sqrt{6}b_0-2b_1\right)+\\ &+\sqrt{6}w^2-2w^3+w^3v(3a_0-2b_0),\\ \frac{dv}{d\tau}=&v^2\left(-32a_0+\sqrt{32}a_1+32b_0-\sqrt{32}b_1\right)+\\ &+wv^2\left(\sqrt{6}a_0-a_1-\sqrt{6}b_0+b_1\right)+\\ &+w^2v^2(-a_0+b_0)-\sqrt{6}6wv+w^2v.\end{aligned} \tag{9.1}$$

Эта система имеет нулевые собственные значения в стационарной точке $w=v=0$, поэтому мы должны снова применить степенное преобразование.

9.1. Подслучай $3a_0 - 2b_0 = b(3a_1 - 2b_1)$

Этот случай является расширением условий $a_0=a_1b$, $b_0=b_1b$ из (7.5). Если $b^2=2/3$, уравнение (9.1) можно переписать так

$$\begin{aligned}\frac{dw}{d\tau} =& -\frac{3v}{2b}\left[(3a_0-2b_0)-b(3a_1-2b_1)\right]+\\ &+wv\left(272a_0-3\sqrt{6}a_1-9b_0+2\sqrt{6}b_1\right)+\\ &+w^2v\left(-9\sqrt{32}a_0+3a_1+3\sqrt{6}b_0-2b_1\right)-\\ &-2w^3+w^3v(3a_0-2b_0)+\sqrt{6}w^2,\\ \frac{dv}{d\tau} =& v^2\left(-32a_0+\sqrt{32}a_1+32b_0-\sqrt{32}b_1\right)+\\ &+wv^2\left(\sqrt{6}a_0-a_1-\sqrt{6}b_0+b_1\right)+\\ &+w^2v^2(-a_0+b_0)-\sqrt{6}6wv+w^2v.\end{aligned} \tag{9.2}$$

Мы видим, что в системах (9.1) и (9.2) коэффициент при v в линейной части первого уравнения равен нулю, если $3a_0 - 2b_0 = b(3a_1 - 2_1)$. В этом подслучае мы используем преобразование

$$u \to w - 1/b,\ v \to rw,\ \dot v \to \dot r w + r\dot w, \tag{9.3}$$

с масштабированием по времени путем деления уравнений на $w/\sqrt{6}$, т.е. $\tilde\tau = w\tau/\sqrt{6}$. Тогда из системы (4.2) при $b^2=2/3$ можно получить

$$\begin{aligned}\frac{dw}{d\tilde\tau} &= 6w+3(3a_1-2b_1)rw-2\sqrt{6}w^2-\\ &-2\sqrt{6}(3a_1-2b_1)rw^2+2(3a_1-2b_1)rw^3,\\ \frac{dr}{d\tilde\tau} =& -7r-(9a_1-\sqrt{32}b_0-5b_1)r^2+3\sqrt{6}rw+\\ &+(7\sqrt{6}a_1-2b_0-13\sqrt{23}b_1)r^2w-\\ &-(8a_1-\sqrt{32}b_0-163b_1)r^2w^2.\end{aligned} \tag{9.4}$$

Это трехпараметрическая система с резонансом 13-го порядка в стационарной точке $w=0, r=0$ на инвариантной прямой $w=0$. Вдоль этой линии есть еще одна стационарная точка. Для нее можно доказать локальную интегрируемость системы, и она не дает никаких дополнительных ограничений на параметры.

Мы вычислили нормальную форму для системы (9.4) до 26-го порядка и получили два уравнения для условия **A**. Это $A_{13}=0$ и $A_{26}=0$, где A_{13} и A_{26} приведены в [19]. Каждое из этих уравнений шестого и двенадцатого порядков однородно по параметрам a_1, b_0 и b_1 системы (3.3). Многочлены A_{13} и A_{26} обнуляются при условиях (7.5).

Однородные алгебраические уравнения с тремя переменными можно переписать как неоднородные уравнения с двумя переменными. Если мы предположим, что $a_1=0$, мы получим только одномерные и нульмерные решения в пространстве параметров. В случае $a_1\neq0$ подставляем $b_0=c_0a_1$, $b_1=c_1a_1$ и получаем систему двух уравнений с двумя переменными $A_{13}(c_0,c_1)=0$, $A_{26}(c_0,c_1)=0$. Результант этих многочленов по каждой из двух переменных тождественно равен нулю, так что достаточно решить только уравнение $A_{13}(c_0,c_1)=0$. Полином A_{13} факторизуется в произведение четырех множителей, включая a_1^6

$$\begin{aligned}A_{13} =& 48\left(c_1-\frac{3}{2}\right)a_1^6\times\left(c_0-\frac{1}{12}\sqrt{6}c_1+\frac{1}{2}\sqrt{6}\right)^2\times\\ &\times\left[409790784c_0^3-104\sqrt{6}c_0^2\times(-9152256+3385633c_1)-\right.\\ &-208c_0\left(-10917702+c_1(-360720+3319927c_1)\right)+\\ &+\sqrt{6}\left(-718439040+c_1(2461047528+\right.\\ &\left.\left.+c_1(-1944898681+441207868c_1)\right)\right)\Big].\end{aligned} \tag{9.5}$$

Из первых двух множителей получаем пару решений $c_1=3/2$ и $c_1=6+2\sqrt{6}c_0$ или

$$\begin{aligned}&5)\ a_0=\left(2b_0+b(3a_1-2b_1)\right)/3,\\ &b_1=3a_1/2,\quad b=\sqrt{2/3},\\ &6)\ a_0=\left(2b_0+b(3a_1-2b_1)\right)/3,\\ &b_1=6a_1+2\sqrt{6}b_0, b=\sqrt{2/3}.\end{aligned} \tag{9.6}$$

Для решений (9.6) мы вычислили нормальную форму системы (9.5) до 36-го порядка и получили для каждого случая линейную систему. Итак, это случаи локальной интегрируемости.

Мы изучили решения двухпараметрическим случаем с коэффициентами над алгебраическим расширением рациональных чисел с алгебраическим числом $\sqrt{6}$. Последний множитель в (9.5) не имеет таких корней. Однако в принципе возможно существование решений в других алгебраических расширениях.

Соответствующая пара первых интегралов системы для найденных параметров (9.6) выписана в Приложении.

9.2. Подслучай $3a_0 - 2b_0 \neq b(3a_1 - 2b_1)$

Если мы воспользуемся преобразованием

$$v \to w^2 p,\quad \dot v \to 2\dot w w p + w^2\dot p \quad\text{и}\quad \tilde\tau = w\tau/\sqrt{6}, \tag{9.7}$$

то получаем из системы (9.1) систему

$$\begin{aligned}\frac{dp}{d\tilde{\tau}} &= -13p - 7\sqrt{6}a_0p^2w^3 + 60a_0p^2w^2 - \\ &-57\sqrt{\frac{3}{2}}a_0p^2w + 27a_0p^2 - 7\sqrt{6}a_1p^2w^2 + \\ &+39a_1p^2w - 9\sqrt{6}a_1p^2 + 5\sqrt{6}b_0p^2w^3 - \\ &-42b_0p^2w^2 + 39\sqrt{\frac{3}{2}}b_0p^2w - 18b_0p^2 + \\ &+5\sqrt{6}b_1p^2w^2 - 27b_1p^2w + 6\sqrt{6}b_1p^2 + 5\sqrt{6}pw, \\ \frac{dw}{d\tilde{\tau}} &= 6w + 3\sqrt{6}a_0pw^4 - 27a_0pw^3 + \\ &+27\sqrt{\frac{3}{2}}a_0pw^2 - \frac{27}{2}a_0pw + 3\sqrt{6}a_1pw^3 - \\ &-18a_1pw^2 + 9\sqrt{\frac{3}{2}}a_1pw - 2\sqrt{6}b_0pw^4 + \\ &+18b_0pw^3 - 9\sqrt{6}b_0pw^2 + 9b_0pw - \\ &-2\sqrt{6}b_1pw^3 + 12b_1pw^2 - 3\sqrt{6}b_1pw - 2\sqrt{6}w^2.\end{aligned} \tag{9.8}$$

Вдоль инвариантной прямой $w=0$ эта система имеет две неподвижных точки с координатами

$$\begin{gathered}w = 0, p = 0 \\ w = 0, p = \frac{13}{3\left(9a_0 - 3\sqrt{6}a_1 - 6b_0 + 2\sqrt{6}b_1\right)}.\end{gathered} \tag{9.9}$$

В начале координат имеем резонанс 19-го порядка, а во второй неподвижной точке имеем систему с резонансом 27-го порядка. Знаменатель в координатах второй точки отличен от нуля из-за условия $3a_0 - 2b_0 \neq b(3a_1 - 2b_1)$.

При резонансе 19-го порядка условие **A** выполняется при обнулении однородного 4-мерного полинома 6-го порядка над алгебраическим расширением целых чисел, см. файл [20]. Он состоит из 84 членов и факторизуется системой MATHEMATICA-11 над алгебраическим расширением $\sqrt{6}$ на два множителя. Первый из них линейный

$$20a_0 + 2\sqrt{6}a_1 + 4b_0 + 3\sqrt{6}b_1. \tag{9.10}$$

Второй множитель – однородный многочлен 5-го порядка. Он включает 5-й порядок каждого отдельного параметра, поэтому он не факторизуется. Итак, подходящим решением условия **A** является лишь корень многочлена (9.7)

$$a_0 = -\left(2\sqrt{6}a_1 + 4b_0 + 3\sqrt{6}b_1\right)/20. \tag{9.11}$$

Во второй стационарной точке имеем резонанс 27-го порядка. Здесь мы должны были вычислить до этого порядка ряды с рациональными коэффициентами по 6 переменным p, w, a_0, a_1, b_0, b_1. Нечисленные знаменатели в коэффициентах возникают из-за рационального сдвига в точке (9.9). Для упрощения мы воспользовались решением условия **A** для 19-го порядка (9.11), т.е. фиксировали a_0 и получили трехпараметрическую систему в дробях с ненулевыми знаменателями. Для упрощения мы ввели новый параметр $r = 6\sqrt{6}a_1 + 12b_0 - \sqrt{6}b_1$, который представляет собой знаменатель сдвига (9.9) при соответствующем a_0, если и заменить

$$b_0 = \frac{1}{12}(-6\sqrt{6}a_1 + \sqrt{6}b_1 + r).$$

После этого мы диагонализовали линейную часть матрицы во второй стационарной точке. В ходе этого процесса мы заработали еще один знаменатель, который обозначили как $h = 130a_1 - 75b_1 - 9\sqrt{6}r$. При помощи этого нового параметра мы исключили параметр b_1, положив $b_1 = 26a_1/15 - h/75 - 3\sqrt{6}r/25$. Сначала мы предположили, что этот знаменатель $h \neq 0$. Затем вычислили нормальную форму до 27-го порядка по 3 параметрам r, h и a_1. В итоге получили в исходных параметрах весьма простое уравнение, выделяющее этот случай

$$\left(2a_1 + 3b_1\right) = 0,$$

после перехода от r, h к исходным параметрам.

Вычисление нормальной формы в этом случае было выполнено при помощи специальной программы на языке Standard Lisp [22]. Для этого потребовалось около 700 000 секунд, или 8 суток на скалярном процессоре с тактовой частотой 3,60 ГГц. При этом оказалось достаточно 8 ГБ ОЗУ оперативной памяти. Получили 8 248 088 членов нормализующего преобразования и 754 члена нормальной формы. Числитель этой нормальной формы представляет собой однородный многочлен 54-го порядка по 3 параметрам. Коэффициенты этого многочлена содержат более чем сто цифр. Этот многочлен факторизуется. Числитель состоит из ненулевого множителя r^{27}, однородного множителя 24-го порядка и квадрата линейного полинома $2a_1 + 3b_1$, см. файл [21]. Прежде чем получить этот результат, мы пытались использовать задержанные вычисления, модулярную арифметику и т.д.

Таким образом, последнее, 7-е решение условия интегрируемости выглядит так

$$\begin{gathered}a_0 = -\left(2\sqrt{6}a_1 + 4b_0 + 3\sqrt{6}b_1\right)/20,\ a_1 = -\frac{3}{2}b_1, \\ 3a_0 - 2b_0 \neq b\left(3a_1 - 2b_1\right),\quad b = \sqrt{2/3}.\end{gathered} \tag{9.12}$$

Случай $h=0$ проще для расчета и представляет собой частный случай условия (9.12)

$$a_0 = -\left(2\sqrt{6}a_1 + 4b_0 + 3\sqrt{6}b_1\right)/20,$$
$$a_1 = -\frac{3}{2}b_1, \qquad (9.13)$$
$$b_0 = -\frac{5a_1}{3\sqrt{6}}, \quad b = \sqrt{2/3}.$$

Седьмое условие похоже на случаи 3) и 5) выше, однако соответствующее множество параметров не пересекается ни с одним из случаев 1) – 6). Соответствующий первый интеграл для 7-го случая приведен в Приложении.

10. ВЫВОДЫ

Для пятипараметрической негамильтоновой вырожденной двумерной системы (4.2) мы изучили возможные необходимые условия интегрируемости. Мы нашли 7 семейств, удовлетворяющих этим условиям. Эти семейства были найдены как двумерные многообразия в параметрическом пространстве. Для каждого из них мы вычислили первый интеграл движения. Для каждого такого интеграла может быть выписано соответствующее решение исследуемой системы в квадратурах.

11. ПРИЛОЖЕНИЕ

Случаи интегрируемости и первые интегралы выглядят следующим образом.

1. При $b_1=0$, $a_0=0$, $a_1=-b_0b$ первый интеграл системы (4.2) выглядит так

$$I_{1uv} = u^2(3b+2u)v^6.$$

Интеграл в переменных исходной системы уравнений (3.3)

$$I_{1xy} = 2x^3 + 3by^2.$$

2. При $b_1=-2a_1$, $a_0=a_1b$, $b_0=-2a_1b$ интеграл для системы (4.2) это

$$I_{2uv} = u^2v^6\left(3b + u\left(2 - 6a_1bv\right)\right).$$

Интеграл в переменных исходной системы (3.3)

$$I_{2xy} = 2x^3 - 6a_1bx^2y + 3by^2.$$

3. При $b_1=(3/2)a_1$, $a_0=a_1b$, $b_0=(3/2)a_1b$ интеграл для системы (4.2)

$$I_{3uv} = [4 - 4a_1uv + 3^{5/6}a_1 \times$$
$$\times {}_2F_1\left(2/3,1/6;\ 5/3;\ -2u/(3b)\right)\times$$
$$\times uv\left(3+2u/b\right)^{1/6}]/[u^{1/3}v(3b+2u)^{1/6}].$$

Этот интеграл в переменных исходной системы уравнений (3.3)

$$I_{3xy} = \left[a_1x^2\left(-4+3^{5/6}{}_2F_1(2/3,1/6;\ 5/3;-2x^3/(3by^2)\times\right.\right.$$
$$\left.\left.\times(3+2x^3/(by^2))^{1/6}\right)+4y\right]/\left[y^{4/3}(3b+2x^3/y^2)^{1/6}\right].$$

Здесь ${}_2F_1(a,b;c;z)$ – гипергеометрическая функция [23].

4. При $b_1=(8/3)a_1$, $a_0=a_1b$, $b_0=(8/3)a_1b$ интеграл для системы (4.2)

$$I_{4uv} = \left[u\left(3+2a_1^2bu\right)+6a_1bv\right]/$$
$$/\left[3u[u^3(6+a_1^2bu)+6a_1^2bu^2v+9bv^2]^{1/6}\right]-$$
$$-8a_1\frac{\sqrt{-b}}{3^{5/3}}B_{6+a_1\sqrt{-6bu}+3v\sqrt{-6b/u^3}}\left(\frac{5}{6},\frac{5}{6}\right).$$

Этот интеграл в переменных исходной системы уравнений (3.3)

$$I_{4xy} = \frac{2a_1^2bx^7+6a_1by^5+3x^4y^2}{3x^4y^2\sqrt[6]{\dfrac{a_1^2bx^{12}}{y^8}+\dfrac{6a_1^2bx^5}{y^3}+\dfrac{9by^2}{x^2}+\dfrac{6x^9}{y^6}}}-$$
$$-8a_1\frac{\sqrt{-b}}{(3^{5/3})}B_{6+a_1\sqrt{-6bx^3/y^2}+3y/x\sqrt{-6by^6/x^9}}\left(\frac{5}{6},\frac{5}{6}\right).$$

Здесь $B_t(a,b)$ – неполная бета-функция [23].

Вычисление интегралов 1–4 описано в работе [4]. Там же обсуждаются их аналитические свойства.

5. При

$$a_0=(2b_0+b(3a_1-2b_1))/3,\ b_1=(3/2)a_1,\ b=\sqrt{2/3}$$

первым интегралом системы (9.4) будет

$$I_{5rw} = \frac{\left(6b_0-3\sqrt{6}a_1\right)w+9a_1-3\sqrt{6}b_0+\dfrac{42}{r}}{w^{7/6}\sqrt[3]{1-\sqrt{\dfrac{2}{3}}w}}-$$
$$-\frac{\sqrt[6]{12}\left(3\sqrt{3}a_1+4\sqrt{2}b_0\right){}_2F_1\left(\dfrac{1}{3},\dfrac{1}{2};\dfrac{3}{2};\dfrac{\sqrt{3/2}}{w}\right)}{\sqrt{w}}$$

Интеграл в переменных исходной системы уравнений (3.3)

$$I_{5xy} = \frac{6x^2\left[\left(2b_0-\sqrt{6}a_1\right)y+14x\right]+42\sqrt{6}y^2}{\left(2x^3+\sqrt{6}y^2\right)^{7/6}}+$$
$$+\frac{\sqrt[3]{2}\left(3\sqrt{6}a_1+8b_0\right){}_2F_1\left(\dfrac{1}{3},\dfrac{1}{2};\dfrac{3}{2};\dfrac{3y^2}{\sqrt{6}x^3+3y^2}\right)}{\sqrt{\dfrac{2x^3}{y^2}+\sqrt{6}}}.$$

6. При

$$a_0=(2b_0+b(3a_1-2b_1))/3,\ b_1=6a_1+2\sqrt{6}b_0,\ b=\sqrt{2/3}$$

первым интегралом системы (9.4) будет

$$I_{6rw}=\left(r^6w^7(3-\sqrt{6}w)^2\right)^{-2a_1-\sqrt{6}b_0}\times$$
$$\times R(r,w)^{-2\left(3a_1+\sqrt{6}b_0\right)},\ \text{где}\ R(r,w)=$$
$$=r\left(\sqrt{6}a_1w-3a_1+3b_0w-3\sqrt{\frac{3}{2}}b_0\right)+1.$$

Соответствующий интеграл в переменных исходной системы уравнений (3.3) это

$$I_{6xy}=\frac{(\sqrt{6}xy^2+2x^4)^6}{x^6y^{14}\left(\frac{x^3}{y^2}+\sqrt{\frac{3}{2}}\right)^7}\times$$
$$\times\left(\frac{2\left(\sqrt{6}a_1+3b_0\right)x^2y}{2x^3+\sqrt{6}y^2}+1\right)^{-\frac{2\left(3a_1+\sqrt{6}b_0\right)}{2a_1+\sqrt{6}b_0}}.$$

7. При значениях параметров

$$a_0=-(2\sqrt{6}a_1+4b_0+3\sqrt{6}b_1)/20,\ a_1=$$
$$=-\frac{3}{2}b_1,\ 3a_0-2b_0\neq b(3a_1-2b_1),\ b=\sqrt{2/3}$$

интеграл в переменных системы (9.8) будет

$$\left(\frac{2(\sqrt{6}a_1+3b_0)x^2y}{2x^3+\sqrt{6}y^2}+1\right)^{-\frac{2(3a_1+\sqrt{6}b_0)}{2a_1+\sqrt{6}b_0}}\times$$
$$\times p^{12}w^{26}(2w(3p(2w(w(w\times(2w(2b_0w-6\sqrt{6}b_0+5b_1)+$$
$$+90b_0-25\sqrt{6}b_1)-60\sqrt{6}b_0+150b_1)+$$
$$+15(9b_0-5\sqrt{6}b_1))-54\sqrt{6}b_0+225b_1)-$$
$$-20(3\sqrt{6}-w(w^2-2\sqrt{6}w+9)))+$$
$$+27(6b_0-5\sqrt{6}b_1)p+90).$$

Интеграл в переменных исходной системы уравнений (3.3) выглядит так

$$I_{7xy}=48b_0x^5y+120b_1x^2y^3+$$
$$+40\sqrt{6}x^3y^2+40x^6+60y^4. \qquad (9.14)$$

СПИСОК ЛИТЕРАТУРЫ

1. *Bruno A.D.* Analytical form of differential equations (I,II) // Trans. Moscow Math. Soc., 1971. V. 25. P. 131–288; 1972. V. 26. P. 199–239.
2. *Bruno A.D.* Local Methods in Nonlinear Differential Equations. Berlin:Springer-Verlag, 1989. 348 p.
3. *Bruno A.D., Edneral V.F.* On the integrability of a planar system of ODEs near a degenerate stationary point // Zap. Nauchn. Semin. Sankt-Peterburgskogo otdeleniya matematicheskogo instituta im. V.A. Steklova RAN (Proc. Sci. Semin. St. Petersburg Branch of the Steklov Mathematical Institute of the Russian Academy of Sciences). 2009. V. 373. P. 34–47.
4. *Edneral V.F., Romanovski V.G.* Calculation of first integrals of a two-dimensional ODE system near a degenerate stationary point by computer algebra tools // Program. Comput. Software. 2011. V. 37. P. 99–103.
5. *Bruno A.D., Edneral V.F., Romanovski V.G.* On New Integrals of the Algaba-Gamero-Garcia System // Proceedings of 19th International Workshop (CASC 2017). Eds. V.P.Gerdt et al. Lecture Notes in Computer Science, Springer, Switzerland, 2017. V. 10490. P. 40–50. doi: 10.1007/978-3-319-66320-3_4 .
6. *Gutnik S.A., Sarychev V.A.* Symbolic-analytic methods for studying equilibrium orientations of a satellite on a circular orbit // Program. Comput. Software. 2021. V. 47. P. 119–123.
7. *Bruno A.D., Edneral V.F.* Normal forms and integrability of ODE systems // Program. Comput. Software. 2006. V. 32. P. 139–144.
8. *Liénard A.* Etude des oscillations entretenues // Revue générale de l'électricité. 1928. V. 23. P. 901–912 and 946–954.
9. *Cherkas L.A.* Conditions for a Liénard equation to have a center, Differential Equations. 1976. V. 12. № 2. P. 292–298.
10. *Edneral V.F.* Integrable Cases of the Polynomial Liénard-type Equation with Resonance in the Linear Part // Mathematics in Computer Science, 2023. V. 17. № 19. doi: 10.1007/s11786-023-00567-6 .
11. *Bautin N.N.* On the number of limit cycles which appear with the variation of the coefficients from an equilibrium point of focus or center type // AMS Transl. Ser. I. 1962. V. 5. P. 396–414.
12. *Bruno A.D., Edneral V.F.* Algorithmic analysis of local integrability // Dokl. Math. 2009. V. 79. № 1. P. 48–52.
13. *Algaba A., Gamero E., Garcia C.* The integrability problem for a class of planar systems // Nonlinearity, 2009. V. 22. P. 395–420.
14. *Bruno A.D.* Power Geometry in Algebraic and Differential Equations // Elsevier Science, Amsterdam, 2000. 348 p.
15. *Edneral V.F., Khanin R.* Application of the resonant normal form to high-order nonlinear odes using MATHEMATICA // Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 2003. V. 502. № 2–3. P. 643–645.

16. *Hilbert D.* Über die Theorie der algebraischen Formen // Mathematische Annalen, 1890. V. 36. P. 473–534.
17. *Malykh M.D.* On Application of M.N. Lagutinski Method to Integration of Differential Equations in Symbolic Form. Part 1 // Discrete and Continuous Models and Applied Computational Science, 2017. V. 25. № 2. P. 103–112.
doi: 10.22363/2312-9735-2017-25-2-103-112 .
18. *Romanovski V.G., Shafer D.S.* The Center and Cyclicity Problems: A Computational Algebra Approach // Birkhüser, Boston, 2009. 330 p.
19. Equations from condition A for the 13th and 26th orders. https://disk.yandex.ru/d/-R1MKEiZz2vWeA..
20. Factorized equation of condition A of the 19th order. https://disk.yandex.ru/i/U5GS1P-WPe8ZFg.
21. Factorized condition A of the 27th order. https://disk.yandex.ru/i/fWHaojf5vM17uA.
22. *Edneral V.F., Khrustalev, O.A.* Package for reducing ordinary differential equations to normal form // Program. Comput. Software. 1992. V. 18. № 5. P. 234–239.
23. *Bateman H., Erdelyi A.* Higher Transcendental Functions, McGraw-Hill, 1953. V. I.

INTEGRATION OF A DEGENERATE SYSTEM OF ODEs

© 2024 A. D. Bruno[a], V. F. Edneral[b]

[a] *Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, 125047 Russia*
[b] *Skobeltsyn Institute of Nuclear Physics, Moscow State University, Moscow, 119991 Russia*

The integrability of a two-dimensional autonomous polynomial system of ordinary differential equations (ODEs) with a degenerate singular point at the origin that depends on six parameters is investigated. The integrability condition for the first quasihomogeneous approximation allows one of these parameters to be fixed on a countable set of values. The further analysis is carried out for this value and five free parameters. Using the power geometry method, the system is reduced to a non-degenerate form through the blowup process. Then, the necessary conditions for its local integrability are calculated using the method of normal forms. In other words, the conditions for the parameters under which the original system is locally integrable near the degenerate stationary point are found. By resolving these conditions, we find seven twoparameter families in the five-dimensional parametric space. For parameter values from these families, the first integrals of the system are found. The cumbersome calculations that occur in the problem under consideration are carried out using computer algebra.

Keywords: degenerate system of ODEs, normal form and power geometry methods

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 33–44*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.422.8

ПОСТРОЕНИЕ КОМПАРТМЕНТАЛЬНЫХ МОДЕЛЕЙ ДИНАМИЧЕСКИХ СИСТЕМ С ПРИМЕНЕНИЕМ ПРОГРАММНОГО КОМПЛЕКСА СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ НА ЯЗЫКЕ JULIA

© 2024 г. А. В. Демидова[a], *, О. В. Дружинина[b], **, О. Н. Масина[c], *, А. А. Петров[c], ******

[a]Российский университет дружбы народов, 117198 Москва, ул. Миклухо-Маклая, д. 6, Россия

[b]Федеральный исследовательский центр "Информатика и управление" Российской академии наук 119333 Москва, ул. Вавилова, д. 44, к. 2, Россия

[c]Елецкий государственный университет им. И. А. Бунина 399770 Елец, Липецкая обл., ул. Коммунаров, д. 28, Россия

**E-mail: demidova-av@rudn.ru,*

***E-mail: ovdruzh@mail.ru,*

****E-mail: olga121@inbox.ru,*

*****E-mail: xeal91@yandex.ru*

Поступила в редакцию 01.08.2023
После доработки: 10.09.2023
Принята к публикации 01.10.2023

В работе рассмотрены вопросы построения компартментальных моделей динамических систем с применением программного комплекса символьных вычислений на языке Julia. Программный комплекс направлен на решение задачи унификации формализованного построения моделей с учетом сущностного описания возможных взаимодействий компартментов и влияния различных факторов на эволюцию систем. Развивается подход к разработке инструментально-методического обеспечения моделирования динамических систем, поведение которых может быть охарактеризовано одношаговыми процессами. Разработанное программное обеспечение позволяет получить символьное представление дифференциальных уравнений модели как в стохастическом, так и в детерминированном случае. Предложенный программный комплекс реализован с помощью языка Julia и использует библиотеку компьютерной алгебры Julia Symbolics. Представлено сравнение инструментария Julia Symbolics с другими системами компьютерной алгебры. Рассмотрено применение разработанного программного комплекса к компартментальной модели распространения эпидемии. Результаты могут найти применение при решении задач конструирования и исследования динамических моделей естествознания, представляемых одношаговыми процессами.

Ключевые слова: компартментальные модели, динамические системы, компьютерная алгебра, язык программирования Julia, программный комплекс символьных вычислений

DOI: 10.31857/S0132347424020051 **EDN:** RPCESS

1. ВВЕДЕНИЕ

Направление, связанное с формализованным представлением моделей динамических систем в символьном виде, относится к числу актуальных научных направлений [1–5]. В частности, теоретический и прикладной интерес представляет построение моделей динамических систем с привлечением методов и средств компьютерной алгебры.

В настоящее время возникает необходимость описания и изучения новых факторов, влияющих на динамику распространения эпидемий [6–10]. Математическому моделированию процессов, связанных с пандемией Covid-19, посвящены, в частности, работы [11–13]. Моделирование эпидемиологических процессов позволяет учесть факторы, влияющие на распространение заболеваний, и оценить действенность различных мер, направленных на снижение риска заболевания. Например, действие таких мер, как дистанцирование, ношение масок, различные протоколы обслуживания в медицинских учреждениях и многое другое, могут быть проанализированы и оптимизированы с помощью эпидемиологических моделей.

Описание различных эффектов в моделях распространения эпидемий связано с повышением размерности, что приводит к достаточно громоздким формализованным представлениям взаимосвязей переменных и параметров. Поэтому целесообразно использовать инструменты компьютерной алгебры для реализации возможностей построения моделей из принципов взаимодействия элементов системы.

Развиваемый в настоящей работе подход к разработке инструментально-методического обеспечения моделирования динамических систем базируется на реализации возможностей построения и исследования систем, поведение которых может быть описано одношаговыми процессами. Как известно, класс одношаговых процессов охватывает многие процессы физики, химии, экологии, демографии, социологии. Большинство моделей для описания процессов распространения эпидемий относится к классу компартментальных моделей, которые являются частным случаем моделей одношаговых процессов.

Для описания компартментальных моделей может быть использован метод построения самосогласованных стохастических моделей [14–17]. В основе этого метода лежит комбинаторная методология [18, 19], в соответствии с которой предполагается, что эволюция многомерной системы рождения–гибели может быть рассмотрена как результат индивидуальных взаимодействий между элементами этой системы. Благодаря указанному методу возможно получить стохастическую модель с согласованной стохастической и детерминистической частями, что позволяет оценить влияние введения стохастики на поведение системы. Метод построения самосогласованных стохастических моделей удобен для дальнейшей разработки программного обеспечения, поскольку позволяет реализовать автоматизированное построение компартментальных моделей по описанию взаимодействий между элементами и по характеру переходов из состояния в состояние.

Одной из важных задач, возникающих при моделировании компартментальных систем, является создание эффективного и удобного для использования программного обеспечения, позволяющего достаточно просто формализовать возможные переходы из состояния в состояние. Разработка таких инструментов компьютерной алгебры, которые являются высокопроизводительными, гибкими, масштабируемыми, предполагает использование широкого спектра технологий программирования. Одним из современных языков программирования, ориентированных на математическое моделирование с высоким уровнем абстракции, является Julia [20]. Данный язык позволяет реализовать алгоритмы символьных вычислений для решения различных задач [21–23].

Настоящая работа посвящена проблеме построения как детерминированных, так и стохастических компартментальных моделей с применением программного комплекса символьных вычислений на языке Julia. Структура статьи следующая. В разделе 2 приводится описание подходов к построению эпидемиологической модели SIR и ее модификаций. В разделе 3 рассмотрены особенности использования языка Julia для решения задач моделирования динамических систем. Представлено сравнение инструментария Symbolics.jl с другими системами компьютерной алгебры. В разделах 4 и 5 изложены основные результаты работы. В разделе 4 дано описание программного комплекса символьных вычислений с учетом различных функциональных модулей комплекса и приведены основные фрагменты кода. В разделе 5 рассмотрено применение разработанного программного комплекса на примере модели распространения эпидемии Covid-19. В разделе 6 приведено обсуждение результатов работы.

2. ОСОБЕННОСТИ И МЕТОДОЛОГИЧЕСКИЕ ОСНОВЫ МОДЕЛИРОВАНИЯ ЭПИДЕМИОЛОГИЧЕСКИХ СИСТЕМ

С целью построения различных классов эпидемиологических моделей обычно используют два подхода: подход, приводящий к построению статистических моделей, и подход, приводящий к построению компартментальных моделей.

Первый подход связан с применением методов математической статистики и машинного обучения для прогнозирования процесса эпидемии в краткосрочной перспективе при отсутствии резких изменений ситуации [24, 25]. Прогнозирование базируется на анализе статистических данных, таких как отчеты о заболевших, умерших и вылечившихся от конкретного заболевания. Второй подход основан на разделении популяции на различные группы (компартменты) по отношению к инфекции и на определении правил перехода из одной группы в другую. При этом процесс распространения эпидемии представляется в виде графа переходов. Реализация таких компартментальных моделей заключается в описании уравнений переходов и последующем анализе динамики численности каждой группы в течение определенного периода времени. В качестве математического аппарата для этого класса моделей чаще всего используются системы дифференциальных или разностных уравнений, однако в последнее время набирает популярность алгоритмическое моделирование, в частности, построение имитационных и агентно-ориентированных моделей [11, 26].

Как известно, при моделировании распространения инфекционных заболеваний существенное значение могут иметь демографические процессы.

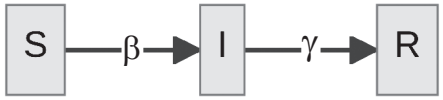


Рис. 1. Диаграмма модели SIR.

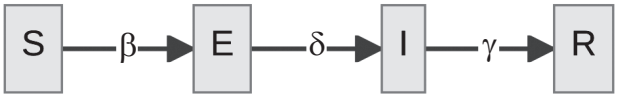


Рис. 2. Диаграмма модели SEIR.

Поэтому при построении модели в зависимости от специфики инфекционной болезни может потребоваться учет таких процессов, как рождаемость, смертность и миграция. Так, например, при моделировании заболеваний с длительным инкубационным или инфекционным периодом (туберкулез, ВИЧ/СПИД) необходимо учитывать рождаемость и смертность. Это связано с тем, что продолжительность болезни сравнима с продолжительностью жизни, и демографические процессы могут оказать большое влияние на распространение инфекции. В свою очередь, при моделировании эпидемий заболеваний с коротким инфекционным периодом (корь, коклюш, грипп) демографическими процессами можно пренебречь [27].

Компартментальные модели удобны для изучения влияния различных факторов, таких как вакцинация, изоляция и т.п., на течение динамики эпидемии в связи с тем, что для учета дополнительного фактора достаточно добавить в граф переходов новый узел и определить его связи с остальными узлами графа.

Классической моделью эпидемии является модель SIR, предложенная В. Кермаком и А. МакКендриком [28]. В этой модели различают три состояния индивидуумов: S – восприимчивый, I – инфицированный, R – выздоровевший (или невосприимчивый). Процессы, характерные для модели SIR, могут быть представлены диаграммой на рис. 1, где β – интенсивность заражения, γ – интенсивность выздоровления.

Система дифференциальных уравнений, соответствующая данной диаграмме, имеет вид:

$$\frac{dS}{dt} = -\beta SI, \quad \frac{dI}{dt} = \beta SI - \gamma I, \quad \frac{dR}{dt} = \gamma I. \qquad (1)$$

Самой известной модификацией модели SIR является модель SEIR, которая учитывает количество бессимптомных переносчиков инфекции E. Диаграмма для данной модели может быть получена добавлением узла E в диаграмму модели SIR. На рис. 2 представлена диаграмма модели SEIR, где β – интенсивность заражения, γ – интенсивность выздоровления, δ – величина, обратная среднему инкубационному периоду заболевания.

В последнее время появилось большое количество работ, в которых рассматриваются модификации модели SIR для моделирования эпидемии COVID-19. Указанные модификации получены усложнением диаграммы SIR с помощью добавления в нее различных узлов. Так, например, в [12] описана такая модель SAPHIRE, которая помимо компартментов S, I и R учитывает дополнительные группы индивидуумов: подвергшиеся воздействию инфекции (E), бессимптомные больные (P), зарегистрированные больные (I), незарегистрированные больные (A) и больные, изолированные в больнице (H).

Другая модель SIDARTHE распространения COVID-19 [13] включает в себя дополнительные группы индивидуумов по отношению к модели SAPHIRE, а именно, больные с опасными для жизни симптомами и бессимптомные больные (разделенные на выявленных и невыявленных).

Таким образом, компартментальные модели могут иметь сколь угодно сложную структуру за счет расширения первоначальной модели добавлением новых узлов. В результате усложнения структуры приобретает актуальность решение задачи формализованного построения таких систем дифференциальных уравнений, которые соответствуют графам рассматриваемых моделей.

В данной работе развивается подход к построению компартментальных моделей в виде систем дифференциальных уравнений для описания процессов распространения эпидемий с возможностью учета разнообразных факторов. В рамках данного подхода реализуется возможность построения как детерминированных, так и стохастических компартментальных моделей.

Развиваемый нами подход к построению компартментальных моделей распространения эпидемий согласуется с методологией построения моделей одношаговых процессов [14, 15]. В рамках этой методологии сначала необходимо сформулировать в терминах одношаговых процессов все взаимодействия между ее элементами системы. Далее необходимо составить такую схему взаимодействия элементов, на основе которой записываются операторы состояния системы и оператор изменения состояния системы. В результате произведения оператора изменения состояния системы и вектора интенсивностей переходов можно получить коэффициенты сноса и диффузии для уравнения Фоккера–Планка. Таким образом, можно получить кроме стохастического описания системы также детерминистическое, которому соответствует вектор сносов.

В следующем разделе мы анализируем такие возможности и особенности языка Julia и системы компьютерной алгебры Symbolics.jl, которые используются для программной реализации описанного подхода к построению компартментальных моделей.

3. ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ЯЗЫКА JULIA ДЛЯ РЕШЕНИЯ ЗАДАЧ МОДЕЛИРОВАНИЯ

Язык программирования Julia представляет собой высокоуровневый интерпретируемый язык с динамической типизацией, который был создан специально для научных вычислений. Одним из важнейших принципов разработки данного языка является обеспечение максимально высокой производительности программного обеспечения, что достигается благодаря использованияю Low Level Virtual Machine (LLVM) в качестве кодогенератора. В ряде работ выполнен сравнительный анализ производительности в научных задачах (см., например, [29]). Результаты проведенного анализа демонстрируют, что скорость выполнения операций с применением языка Julia сравнима с наиболее высокопроизводительными математическими пакетами и языками программирования.

Следует отметить, что уровень выразительности языка по отношению к задачам математического моделирования является достаточно высоким. По сравнению с классическими объектно-ориентированными языками (такими, как Python, C++, Java) Julia основана на концепции множественной диспетчеризации. Концепция характеризуется тем, что главными объектами являются так называемые мультиметоды, а система типов образует иерархическую структуру. По сравнению с Python, где математические операции над тензорами требуют наличия дополнительных библиотек типа Numpy, матрицы и векторы в Julia являются элементами базовой иерархии типов. Благодаря этому факту существует возможность достаточно гибко определять логику программы.

Развитые средства метапрограммирования в Julia позволили создать удобные и высокопроизводительные средства численных методов, математического моделирования и компьютерной алгебры. Пакет DifferentialEquations.jl на сегодняшний день является один из наиболее функциональных пакетов для численного решения дифференциальных уравнений. В частности, поддерживаются функции решения следующих типы уравнений:

- дискретные уравнения (функциональные карты, дискретные стохастические (Gillespie/Markov) симуляции);
- обыкновенные дифференциальные уравнения (ODEs);
- сплит и разделенные ODE (симплектические интеграторы, методы IMEX);
- стохастические обыкновенные дифференциальные уравнения (SODE или SDE);
- стохастические дифференциально-алгебраические уравнения (SDAE);
- случайные дифференциальные уравнения (RODE или RDE);
- дифференциальные алгебраические уравнения (DAEs);
- дифференциальные уравнения с запаздыванием (DDE);
- нейтральные, отстающие и алгебраические дифференциальные уравнения с запаздыванием (NDDE, RDDE и DDAE);
- стохастические дифференциальные уравнения с запаздыванием (SDDE);
- экспериментальная поддержка стохастических нейтральных, отстающих и алгебраических дифференциальных уравнений с запаздыванием (SNDDE, SRDDEs и SDDAEs);
- смешанные дискретные и непрерывные уравнения (гибридные уравнения) Jump Diffusions;
- детерминированные и стохастические дифференциальные уравнения в частных производных.

Для DifferentialEquations.jl поддерживается широкая интеграция с возможностями других пакетов, в частности ускорение на специализированных процессорах (GPU), автоматическое детектирование разреженности, символьное вычисления якобианов, интерполяция решений. Возможности пакета более подробно рассмотрены, например, в [30]. Средства символьной компьютерной алгебры в языке Julia реализуются посредством пакета Symbolics.jl. Высокопроизводительным символьным вычислениям в Julia посвящена работа [21]. Существует ряд работ, посвященных применению языка Julia при моделировании процессов в различных областях знания. В частности, вопросам биохимического моделирования в Julia средствами Catalyst.jl посвящена работа [31].

Следует отметить, что разработан ряд библиотек, направленных на решение прикладных вопросов математического моделирования. В частности, в [32] рассмотрены вопросы моделирования распространения патогенов и вопросы разработки специализированной библиотеки Pathogen.jl.

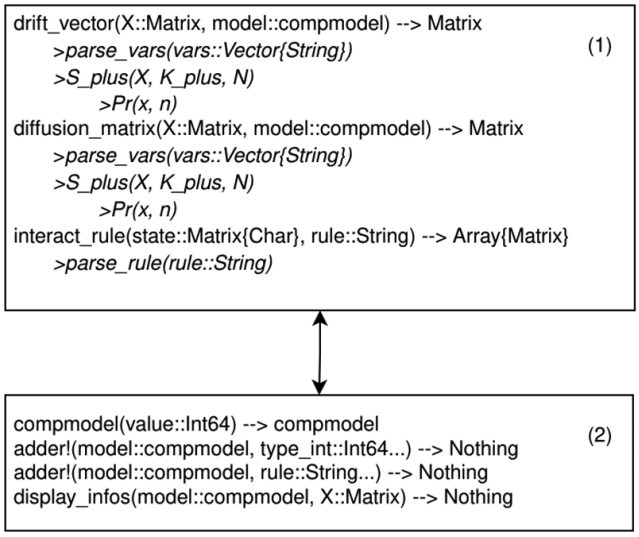


Рис. 3. Структура программного комплекса.

Вопросы применения численно-аналитических методов для моделирования систем, задаваемых дифференциальными уравнениями, с применением языка Julia изложены а работе [23]. Язык Julia может быть использован для моделирования сложных технических систем [33, 34].

Сравнение Symbolics.jl с другими известными системами компьютерной алгебры представлено в табл. 1.

Следует отметить, что основными преимуществами Symbolics.jl являются высокая производительность, открытость и простота интеграции благодаря возможности генерации кода для языка C. Выразительность языка Julia по отношению к научным задачам и развитые возможности метапрограммирования предоставляют удобные инструменты для решения задач формализации компартментальных моделей в символьной форме.

4. ОПИСАНИЕ ПРОГРАММНОГО КОМПЛЕКСА СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ

В данном разделе мы приводим описание программного комплекса символьных вычислений, который позволяет получать детерминистическое и стохастическое описание модели по схеме взаимодействий ее элементов. Разработка указанного программного комплекса осуществлена на языке Julia с применением библиотеки Symbolics.jl, а также результатов [5].

Программный комплекс состоит из конструирующего и функционального блоков. Конструирующий блок определяет структуру модели. Указанная структура включает в себя размерность фазового пространства системы, матрицы переходов, счетчик коэффициентов интенсивности переходов, текстовое представление коэффициентов интенсивности переходов, операторы состояний M, N и оператор переходов r. Фрагмент программного кода, описывающего структуру модели и функцию-конструктор, представлен в листинге 1.

```
using Symbolics
""" value - количество компартментов
 ↪  """
mutable struct compmodel
    value::Int64
    M::Union{Vector,Matrix}
    N::Union{Vector,Matrix}
    eq_nub::Int64
    coef::Array{String}
    interaction::Matrix{Int64}
end
compmodel(value::Int64) =
 ↪  compmodel(value, zeros(value),
 ↪  zeros(value),  0, [],
 ↪  Array{Any}(undef, 0, 0))
```

Листинг 1. Программный код структуры модели.

Кроме того, конструирующий блок включает в себя связанные со структурой модели функции, которые задают и отображают схему взаимодействий. Указанным функциям соответствуют метод display_infos и мультиметод adder!.

Метод display_infos отвечает за вывод правил переходов и может быть использован для проверки

Таблица 1. Сравнение систем компьютерной алгебры

CAS / Особенности	Цена и лицензия	Быстродействие	Сложность интеграции	Функциональность
Symbolics.jl	Бесплатно/ открытая	Высокое	Невысокая	Средняя
SymPy	Бесплатно/ открытая	Низкое	Невысокая	Высокая
Maxima	Бесплатно/ открытая	Высокое	Высокая	Высокая
Symbolic Math Toolbox (MATLAB)	Варьируется/ закрытая	Высокое	Невысокая	Высокая
Mathematica	Варьируется/ закрытая	Высокое	Невысокая	Высокая

корректности работы программного комплекса. Мультиметод adder! позволяет добавлять в систему правила переходов двумя способами. Первый способ основан на задании схемы взаимодействий в виде предварительно определенного вектора и номеров компартментов. Второй способ базируется на задании схемы взаимодействий в символьном виде. Заголовки функций метода adder! представлены в листингах 2, 3.

```
function adder!(model::compmodel,
↪  type_int::Int64,
  id_pop_1::Int64, id_pop_2::Int64,
  coef = nothing,
  ↪  interact::Union{Nothing,Matrix} =
  ↪  nothing)
```

Листинг 2. Заголовок метода adder! с использованием таблицы взаимодействий.

```
function adder!(model::compmodel,
↪  rule::String,
  state::Matrix, coef::String)
```

Листинг 3. Заголовок метода adder! с использованием правил.

В листинге 2 представлен заголовок метода, получающего следующие аргументы: модель, тип взаимодействия (задается таблицей взаимодействий), номера компартментов id_pop_1 и id_pop_2, коэффициент интенсивности перехода в символьной форме. Отметим, что по умолчанию коэффициент назначается из диапазона $k_1, k_2, ..., k_n$, где n – количество правил переходов.

В листинге 3 представлен заголовок метода, получающего следующие аргументы: модель, тип взаимодействий в символьной форме, фазовые переменные в символьной форме, коэффициент интенсивности переходов. Следует отметить, что обе реализации мультиметода являются in-place, т.е. изменяют состояние модели, при этом возвращается значение “nothing”.

Функциональный блок программного комплекса содержит в себе функции, которые предназначены для построения схемы взаимодействия на основе описания, а также для вывода вектора сноса и матрицы диффузии.

Основным методом блока является метод interact_rule, который преобразует текстовое описание правила в форме

“a + b ~ c + d” или “a ~ c”

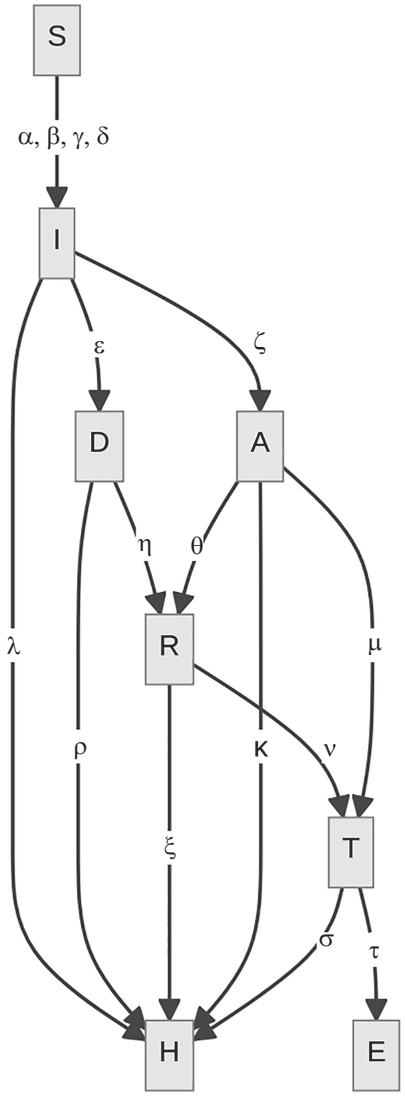


Рис. 4. Диаграмма модели SIDARTHE.

в векторы M, N для соответствующей модели. Указанный метод использует вспомогательный метод parse_rule, который разбирает текстовое правило на антецеденты и консеквенты.

Методы drift_vector и duffusion_matrix отвечают за расчет вектора сноса и матрицы диффузии соответственно. Указанные методы принимают в качестве аргументов модель и вектор состояний в символьной форме.

Кроме того, в функциональном блоке присутствуют служебные методы S_plus, Pr и parse_vars. Методы S_plus, Pr реализуют математические операции для методов drift_vector и diffusion_matrix. Метод parse_vars отвечает за преобразования коэффицентов в текстовой форме в символьные переменные.

Общая структура разработанного программного комплекса представлена на рис. 3.

На рис. 3 через (1) обозначен функциональный блок программного комплекса, через (2) обозначен конструирующий блок программного комплекса. Приведены заголовки разработанных методов с возвращаемыми типами. Вызовы служебных методов отмечены курсивным текстом. Диаграмма на рис. 3 приведена для пояснения функциональных возможностей разработанного программного комплекса.

Таким образом, программный комплекс позволяет формализовать компартментальные модели в символьной форме по описанию взаимодействий между элементами. Далее рассмотрим пример построения компартментальной модели с применением разработанного программного обеспечения.

5. ПРИМЕР ПОСТРОЕНИЯ КОМПАРТМЕНТАЛЬНОЙ МОДЕЛИ

Для демонстрации возможностей реализованного программного обеспечения предлагается рассмотреть модель SIDARTHE распространения эпидемии Covid-19 [13]. Выбор данной модели обусловлен тем, что в ней учитывается достаточно много дополнительных факторов по сравнению с моделью SIR. Для модели SIDARTHE известна система дифференциальных уравнений, что позволяет провести верификацию результатов.

В модели SIDARTHE рассматриваются следующие компартменты:

- S – восприимчивый (незараженный);
- I – невыявленный бессимптомный или малосимптомный инфицированный;
- D – диагностированный бессимптомный инфицированный;
- A – невыявленный с симптоматическим инфицированием;
- R – диагностированный с симптоматическим инфицированием;
- T – диагностированный и госпитализированный с опасными для жизни симптомами;
- H – выздоровевший инфицированный;
- E – умерший инфицированный.

Данная компартментальная модель учитывает следующие процессы: заражение; диагностирование заболевания; появление симптомов у инфицированных; госпитализация; выздоровление; летальный исход.

Заражение восприимчивого индивидуума (S) в рассматриваемой модели может произойти в результате его взаимодействия с любым типом инфицированных индивидуумов, кроме госпитализированных индивидуумов. В частности, инфицирование происходит с коэффициентом α после взаимодействия с индивидуумом из компартмента I, с коэффициентом β – с индивидуумом из компартмента D, с коэффициентом γ – с индивидуумом из компартмента A и с коэффициентом δ – с индивидуумом из компартмента R. Выявление инфицированного индивидуума происходит с коэффициентом ε из компартмента I и с коэффициентом θ из компартмента A.

В данной модели предусмотрен учет проявления симптомов болезни с коэффициентом ζ у невыявленных инфицированных I и с коэффициентом η у выявленных инфицированных (D). В данной модели предусмотрено проявление у индивидуумов из компартмента A с коэффициентом μ и из компартмента R с коэффициентом ν симптомов, угрожающих жизни, с последующей госпитализацией индивидуумов.

Кроме того, в рассматриваемой модели учитывается выздоровление инфицированных из всех компартментов: из компартмента I с коэффициентом λ, из A – с коэффициентом κ, из D – с коэффициентом ρ, из R – с коэффициентом ξ, из T – с коэффициентом σ. Наконец, в данной модели учитывается случай возможного летального исхода тяжело больных индивидуумов (T) с коэффициентом τ.

Модели SIDARTHE соответствует диаграмма, представленная на рис. 4.

Система дифференциальных уравнений рассматриваемой модели предложена в [13] и имеет вид:

$$\begin{aligned}
\frac{dS}{dt} &= -S(\alpha I + \beta D + \gamma A + \delta R),\\
\frac{dI}{dt} &= S(\alpha I + \beta D + \gamma A + \delta R) - (\varepsilon + \zeta + \lambda)I,\\
\frac{dD}{dt} &= \varepsilon I - (\eta + \rho)D,\\
\frac{dA}{dt} &= \zeta I - (\theta + \mu + \kappa)A,\\
\frac{dR}{dt} &= \eta D + \theta A - (\nu + \xi)R,\\
\frac{dT}{dt} &= \mu A + \nu R - (\sigma + \tau)T,\\
\frac{dH}{dt} &= \lambda I + \rho D + \kappa A + \xi R + \sigma T,\\
\frac{dE}{dt} &= \tau T.
\end{aligned} \tag{2}$$

Рассмотрим пример использования программного комплекса для формализации модели SIDARTHE. Формализация производится с исполь-

зованием графа, представленного на рис. 4, и с учетом описания процессов, происходящих в рассматриваемой системе. Для формализации необходимо задать количество компартментов ($n = 8$), символьные обозначения компартментов [‘s’ ‘i’ ‘d’ ‘a’ ‘r’ ‘t’ ‘h’ ‘e’] с помощью переменной state_sidarthe, список правил переходов между компартментами с помощью переменной rules_sidarthe и коэффициенты модели с помощью переменной coefs_sidarthe. Само построение модели производится с помощью функции

```
map((rule,coef) -> adder!(sidarthe,
 ↪  rule, state_sidarthe, coef),
 ↪  rules_sidarthe, coefs_sidarthe).
```

Программный код представлен в листинге 4.

```
sidarthe = compmodel(8)

@variables s i d a r t h e
matrix_sidarthe = [s i d a r t h e]
state_sidarthe = ['s' 'i' 'd' 'a' 'r'
 ↪  't' 'h' 'e']
rules_sidarthe = ["s + i ~ i + i",
"s + d ~ i + d", "s + a ~ i + a",
"s + r ~ i + r",
"i ~ a", "i ~ d", "i ~ h",
"a ~ r", "a ~ t", "a ~ h",
"d ~ r", "d ~ h",
"r ~ t", "r ~ h",
"t ~ e", "t ~ h"]
 coefs_sidarthe = ['α', 'β', 'γ',
  ↪  'δ', 'ζ', 'ϵ', 'λ', 'θ', 'μ',
  ↪  'κ', 'η', 'ρ', 'ν', 'ξ', 'τ',
  ↪  'σ']

map((rule,coef) -> adder!(sidarthe,
 ↪  rule, state_sidarthe, coef),
rules_sidarthe, coefs_sidarthe)
```

Листинг 4. Реализация модели.

Для получения вектора сносов можно использовать метод drift_vector. Вектор сносов для системы (2) имеет вид, представленный на рис. 5.

Таким образом, полученный вектор сносов с точностью до обозначений соответствует правой части детерминированной системы дифференциальных уравнений (2).

Кроме того, с помощью метода diffusion_matrix можно получить матрицу диффузии уравнения Фоккера–Планка, что, в свою очередь, позволяет записать стохастическое дифференциальное уравнение в форме Ланжевена. Результат работы программного комплекса применительно к получению матрицы диффузии здесь не приводится ввиду достаточно громоздкого представления.

```
1 drift_vector(matrix_sidarthe, sidarthe)
```

$$\begin{bmatrix} -as\gamma - ds\beta - is\alpha - rs\delta \\ as\gamma - i\varepsilon + ds\beta - i\zeta - i\lambda + is\alpha + rs\delta \\ i\varepsilon - d\eta - d\rho \\ i\zeta - a\theta - a\kappa - a\mu \\ a\theta + d\eta - r\nu - r\xi \\ a\mu + r\nu - t\sigma - t\tau \\ a\kappa + d\rho + i\lambda + r\xi + t\sigma \\ t\tau \end{bmatrix}$$

Рис. 5. Вектор сносов для модели SIDARTHE.

6. ОБСУЖДЕНИЕ

Таким образом, рассмотренный в разделе 5 пример демонстрирует возможности разработанного программного комплекса для построения компартментальных моделей с большим количеством переменных и с достаточно сложной структурой. Результаты конструирования модели SIDARTHE в точности соответствуют формальному описанию в виде системы дифференциальных уравнений (2).

Подход к построению правил взаимодействий в системе продемонстрирован в листинге 6. Мы предлагаем два типа правил, описывающих эволюцию системы, а именно, бинарные и небинарные правила. Конкретная формулировка правил для системы SIDARTHE содержится в переменной rules_sidarthe. Представление правил осуществлено в текстовом формате, что позволяет манипулировать содержимым базы правил с использованием штатных средств языка Julia. Разработанный программный комплекс продемонстрировал возможности удобного инструментария для конструирования компартментальных моделей на основе сущностного описания взаимодействий элементов.

Перспективой развития работы является реализация возможности численного расчета траекторий компартментальных систем для детерминистического и стохастического случая. Реализация такой возможности позволит выполнять комплекс исследований, связанных с качественным поведением систем. Кроме того, в дальнейшем целесообразно разработать модуль программного комплекса для усовершенствования визуализации результатов.

7. ЗАКЛЮЧЕНИЕ

В современных условиях быстрого изменения течения эпидемий и пандемий актуальной задачей является разработка программного обеспечения для оперативного и удобного получения таких новых математических моделей распространения заболеваний, которые учитывают дополнительные специфические факторы.

В настоящей работе представлены результаты разработки программного комплекса символьных вычислений для построения детерминированных и стохастических моделей динамических систем. Программный комплекс направлен на решение задачи формализации компартментальных моделей исходя из сущностного описания взаимодействия компартментов. Использование функциональности разработанного программного обеспечения позволяет избежать некорректностей технического характера при построении моделей высоких размерностей. Применение языка Julia дало возможность привлечь удобные и высокопроизводительные средства численных методов, математического моделирования и компьютерной алгебры.

Результаты могут найти применение при решении задач конструирования и исследования динамических моделей одношаговых процессов, в частности, характерных для биологических, экологических и социо-экономических систем.

СПИСОК ЛИТЕРАТУРЫ

1. *Кулябов Д.С.* Аналитический обзор систем символьных вычислений // Вестник РУДН. Сер. Математика. Информатика. Физика. 2007. № 1–2. С. 38–45.
2. *Колесов Ю.Б., Сениченков Ю.Б.* Компонентное моделирование сложных динамических систем. — СПб: Санкт-Петербургский политехнический университет Петра Великого, 2020.
3. *Банщиков А.В., Бурлакова Л.А., Иртегов В.Д., Титоренко Т.Н.* Символьные вычисления в моделировании и качественном анализе динамических систем // Вычислительные технологии. 2014. № 6. С. 3–18.
4. *Banshchikov A., Vetrov A.* Application of software tools for symbolic description and modeling of mechanical systems // CEUR Workshop Proceedings. 2. ser. "ICCSDE 2020 – Proceedings of the 2nd International Workshop on Information, Computation, and Control Systems for Distributed Environments". 2020. P. 33–42.
5. *Демидова А.В., Дружинина О.В., Масина О.Н., Петров А.А.* Разработка алгоритмического и программного обеспечения моделирования управляемых динамических систем с применением символьных вычислений и стохастических методов // Программирование. 2023. № 2. С. 54–68.
6. *Кабанихин С.И., Криворотько О.И.* Математическое моделирование эпидемии Уханьского коронавируса COVID-2019 и обратные задачи // Журнал вычислительной математики и математической физики. 2020. Т. 60. № 11. С. 1950–1961.
7. *Hamelin F., Iggidr A., Rapaport A., Sallet G.* Observability, Identifiability and Epidemiology A survey. 2023. Access mode: https://arxiv.org/abs/2011.12202.
8. *Chebotaeva V., Vasquez P.A.* Erlang-Distributed SEIR Epidemic Models with Cross- Diffusion // Mathematics. 2023. Vol. 11, no. 9. P. 2167. Access mode: https://www.mdpi.com/2227-7390/11/9/2167.
9. *Киселевская-Бабинина В.Я., Романюха А.А., Санникова Т.Е.* Математическая модель течения Covid-19 и прогноз тяжести инфекции // Математическое моделирование. 2023. Т. 35. № 5. С. 31–46.
10. *Ghosh S., Volpert V., Banerjee M.* An Epidemic Model with Time Delay Determined by the Disease Duration // Mathematics. 2022. Vol. 10, no. 15. P. 2561. Access mode: https://www.mdpi.com/2227-7390/ 10/15/2561.
11. *Ariffin M., Gopal K., Krishnarajah I., Cheilias I., Adam M., Arasan J., Rahman N., Dom N., Sham N.* Mathematical epidemiologic and simulation modelling of first wave COVID-19 in Malaysia // Scientific Reports. 2021. Vol. 11. P. 20739.
12. *Roman H.E., Croccolo F.* Spreading of Infections on Network Models: Percolation Clusters and Random Trees // Mathematics. 2021. Vol. 9, no. 23. P. 3054. Access mode: https://www.mdpi.com/2227-7390/9/23/3054.
13. *Giordano G., Blanchini F., Bruno R., Colaneri P., Filippo A., Di Matteo A., Colaneri M.* Modelling the COVID-19 epidemic and implementation of population-wide interventions in Italy // Nature Medicine. 2020. Vol. 26. P. 855–860.
14. *Демидова А.В.* Уравнения динамики популяций в форме стохастических дифференциальных уравнений // Вестник РУДН. Серия: Математика. Информатика. Физика. 2013. № 1. С. 67–76. Режим доступа: https://journals.rudn.ru/miph/ article/view/8319.
15. *Gevorkyan M.N., Velieva T.R., Korolkova A.V., Kulyabov D.S., Sevastyanov L.A.* Stochastic Runge–Kutta Software Package for Stochastic Differential Equations // Dependability Engineering and Complex Systems / ed. by Zamojski W., Mazurkiewicz J., Sugier J., Walkowiak T., and Kacprzyk J. — Cham : Springer International Publishing. — 2016. Vol. 470. P. 169–179.
16. *Gevorkyan M.N., Demidova A.V., Korolkova A.V., Kulyabov D.S.* Issues in the Software Implementation of Stochastic Numerical Runge–Kutta // Distributed Computer and Communication Networks / ed. by Vishnevskiy V. M. and Kozyrev D. V. Cham : Springer International Publishing. 2018. Vol. 919. P. 532–546.

17. *Геворкян М.Н., Демидова А.В., Велиева Т.Р., Королькова А.В., Кулябов Д.С., Севастьянов Л.А.* Реализация метода стохастизации одношаговых процессов в системе компьютерной алгебры // Программирование. 2018. № 2. С. 18–27.
18. *Gardiner C.W.* Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences. Heidelberg: Springer, 1985.
19. *Van Kampen N.* Stochastic Processes in Physics and Chemistry. Amsterdam: Elsevier, 1992.
20. *Bezanson J., Karpinski S., Shah V., Edelman A.* Julia: A Fast Dynamic Language for Technical Computing. 2012. Access mode: https://arxiv.org/abs/1209.5145.
21. *Gowda S., Ma Y., Cheli A., Gwozzdz M., Shah V.B., Edelman A., Rackauckas C.* High-Performance Symbolic-Numerics via Multiple Dispatch // ACM Commun. Comput. Algebra. 2022. jan. Vol. 55, no. 3. P. 92–96. Access mode: https: //doi.org/10.1145/3511528.3511535.
22. *Кулябов Д.С., Королькова А.В.* Компьютерная алгебра на Julia // Программирование. 2021. № 2. С. 44–50.
23. *Fedorov A.V., Masolova A.O., Korolkova A.V., Kulyabov D.S.* Application of a numerical-analytical approach in the process of modeling differential equations in the Julia language // Journal of Physics: Conference Series. 2020. dec. Vol. 1694, no. 1. P. 012026. Access mode: https://dx.doi. org/10.1088/1742-6596/1694/1/012026.
24. *Abotaleb M.S., Makarovskikh T.* Analysis of Neural Network and Statistical Models Used for Forecasting of a Disease Infection Cases // 2021 International Conference on Information Technology and Nanotechnology (ITNT). 2021. P. 1–7.
25. *Tuluri F., Remata R., Walters W.L., Tchounwou P.B.* Application of Machine Learning to Study the Association between Environmental Factors and COVID-19 Cases in Mississippi, USA // Mathematics. 2022. Vol. 10, no. 6. P. 850. Access mode: https://www.mdpi.com/2227-7390/ 10/6/850.
26. *Roman H.E., Croccolo F.* Spreading of Infections on Network Models: Percolation Clusters and Random Trees // Mathematics. 2021. Vol. 9, no. 23. P. 3054. Access mode: https://www.mdpi.com/2227-7390/ 9/23/3054.
27. *Романюха А.А.* Математические модели в иммунологии и эпидемиологии инфекционных заболеваний. Москва : Бином. Лаборатория знаний, 2011.
28. *Kermack W.O., McKendrick A.G.* Contributions to the mathematical theory of epidemics // Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character. 1927. V. 115. P. 700–721.
29. *Strauss R.R., Bishnu S., Petersen M.R.* Comparing the Performance of Julia on CPUs versus GPUs and Julia-MPI versus Fortran-MPI: a case study with MPASOcean (Version 7.1) // EGUsphere. 2023. Vol. 2023. P. 1–22. Access mode: https://egusphere.copernicus. org/preprints/2023/egusphere-2023-57/.
30. *Rackauckas C., Nie Q.* DifferentialEquations. jl — A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia // Journal of Open Research Software. 2017.
31. *Loman T.E., Ma Y., Ilin V., Gowda S., Korsbo N., Yewale N., Rackauckas C., Isaacson S.A.* Catalyst: Fast Biochemical Modeling with Julia. 2022. Access mode: https://www.biorxiv.org/content/early/2022/08/02/2022.07.30.502135.
32. *Angevaare J., Feng Z., and Deardon R.* Pathogen.jl: Infectious Disease Transmission Network Modeling with Julia // Journal of Statistical Software. 2022. Vol. 104, no. 4. P. 1–30. Access mode: https://www.jstatsoft.org/index.php/ jss/article/view/v104i04.
33. *Апреутесей А.М.Ю., Королькова А.В., Кулябов Д.С.* Возможности гибридного моделирования систем с управлением на языках Modelica и Julia // Распределенные компьютерные и телекоммуникационные сети: управление, вычисление, связь (DCCN–2020). Материалы XXIII Международной научной конференции. 2020. С. 433–440.
34. *Apreutesey A.M.Y., Korolkova A.V., Kulyabov D.S.* Hybrid modelling of the red algorithm in the Julia language // Journal of Physics: Conference Series. 7. Information Technology, Telecommunications and Control Systems, ITTCS 2020. 2020. P. 012025.

CONSTRUCTING COMPARTMENTAL MODELS OF DYNANIC SYSTEMS USING A SOFTWARE PACKAGE FOR SYMBOLIC COMPUTATION IN JULIA

© 2024 A. V. Demidova[a], O. V. Druzhinina[b], O. N. Masina[c], A. A. Petrov[c]

[a] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russia*

[b] *Federal Research Center "Computer Science and Control" of Russian Academy of Sciences 44-2, Vavilov St., Moscow, 119333 Russia*

[c] *Bunin Yelets State University, 28, Kommunarov St., Yelets, Lipetsk region, 399770 Russia*

This paper considers the problem of constructing compartmental models of dynamic systems by using a software package for symbolic calculation written in Julia. The software package is aimed at unifying the formalized construction of compartmental models, taking into account the meaningful description of possible interactions among compartments and the influence of various factors on the evolution of systems. An approach to the development of the instrumental and methodological basis for modeling the dynamic systems the behavior of which can be described by one-step processes is developed. The proposed software package enables the symbolic representation of the differential equations of the model in both stochastic and deterministic cases. It is implemented in Julia and uses the Julia Symbolics computer algebra library. A comparison between the Julia Symbolics tools and some other computer algebra systems is carried out. The application of the developed software package to a compartmental model is considered. The results can be used to solve problems of constructing and studying dynamic models in natural sciences that are represented by onestep processes.

Keywords: compartmental models, dynamic systems, computer algebra, Julia programming language, symbolic computing software package

REFERENCES

1. *Kulyabov D.S.* Analytical overview of symbolic computation systems, Vestn. Ross. Univ. Druzhby Nar., Ser. Mat. Inf. Fiz. 2007. №. 1–2. P. 38–45.
2. *Kolesov Yu.B., Senichenkov Yu.B.* Komponentnoe modelirovanie slozhnykh dinamicheskikh sistem (Component Modeling of Complex Dynamic Systems), St. Petersburg: S.-Peterb. Politekh. Univ. Petra Velikogo, 2020.
3. *Banshchikov A.V., Burlakova L.A., Irtegov V.D., Titorenko T.N.* Symbolic computation in modeling and qualitative analysis of dynamic systems // Vychisl. Tekhnol. 2014. № 6. P. 3–18.
4. *Banshchikov A., Vetro A.* Application of software tools for symbolic description and modeling of mechanical systems, Proc. 2nd Int. Workshop Information, Computation, and Control Systems for Distributed Environments (ICCSDE). 2020. P. 33–42.
5. *Demidova A.V., Druzhinina O.V., Masina O.N., Petrov A.A.* Development of algorithms and software for modeling controlled dynamic systems using symbolic computations and stochastic methods // Program. Comput. Software. 2023. V. 49. P. 108–121.
6. *Kabanikhin S.I., Krivorotko O.I.* Mathematical modeling of the Wuhan COVID-2019 epidemic and inverse problems // Comput. Math. Math. Phys. 2020. V. 60. P. 1889–1899.
7. *Hamelin F., Iggidr A., Rapaport A., Sallet G.* Observability, identifiability, and epidemiology: A survey, 2023. https://arxiv.org/abs/2011.12202.
8. *Chebotaeva V., Vasquez P.A.* Erlang-distributed SEIR epidemic models with cross-diffusion, Mathematics. 2023. V. 11. № 9. P. 2167. https://www.mdpi.com/2227-7390/11/9/2167.
9. *Kisselevskaya-Babinina V.Ya., Romanyukha A.A., Sannikova T.E.* Mathematical model of COVID-19 progression: Prediction of severity and outcome, Math. Models Comput. Simul. 2023. V. 15. P. 987–998.
10. *Ghosh S., Volpert V., Banerjee M.* An epidemic model with time delay determined by the disease duration // Mathematics. 2022. V. 10. № 15. P. 2561. https://www.mdpi.com/2227-7390/10/15/2561.
11. *Ariffin M., Gopal K., Krishnarajah I., Cheilias I., Adam M., Arasan J., Rahman N., Dom N., Sham N.* Mathematical epidemiologic and simulation modeling of first wave COVID-19 in Malaysia // Sci. Rep. 2021. V. 11. P. 20739.
12. *Roman H.E., Croccolo F.* Spreading of infections on network models: Percolation clusters and random trees, Mathematics. 2021. V. 9. № 23. P. 3054. https://www.mdpi.com/2227-7390/9/23/3054.
13. *Giordano G., Blanchini F., Bruno R., Colaneri P., Filippo A., Di Matteo A., Colaneri M.* Modeling the COVID-19 epidemic and implementation of population-wide interventions in Italy // Nat. Med. 2020. V. 26. P. 855–860.
14. *Demidova A.V.* Equations of population dynamics in the form of stochastic differential equations // Vestn. Ross. Univ. Druzhby Nar., Ser. Mat. Inf. Fiz. 2013. № 1. P. 67–76. https://journals.rudn.ru/miph/article/view/8319.
15. *Gevorkyan M.N., Velieva T.R., Korolkova A.V., Kulyabov D.S., Sevastyanov L.A.* Stochastic Runge–Kutta software package for stochastic differential equations // Dependability Engineering and Complex Systems, Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., and Kacprzyk, J., Eds., Springer. 2016. V. 470. P. 169–179.

16. *Gevorkyan M.N., Demidova A.V., Korolkova A.V., Kulyabov D.S.* Issues in the software implementation of stochastic numerical Runge–Kutta // Distributed Computer and Communication Networks, Vishnevskiy, V.M. and Kozyrev, D., Eds., Springer. 2018. V. 919. P. 532–546.
17. *Gevorkyan M.N., Demidova A.V., Velieva T.R., Korolkova A.V., Kulyabov D.S., Sevastyanov L.A.* Implementing a method for stochastization of one-step processes in a computer algebra system, Program. Comput. Software. 2018. V. 44. P. 86–93.
18. *Gardiner C.W.* Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences, Heidelberg: Springer, 1985.
19. *Van Kampen N.* Stochastic Processes in Physics and Chemistry, Amsterdam: Elsevier, 1992.
20. *Bezanson J., Karpinski S., Shah, V., Edelman A.* Julia: A fast dynamic language for technical computing, 2012. https://arxiv.org/abs/1209.5145.
21. *Gowda S., Ma Y., Cheli A., Gwozzdz M., Shah V.B., Edelman A., Rackauckas C.* High-performance symbolic-numerics via multiple dispatch // ACM Commun. Comput. Algebra. 2022. V. 55. № 3. P. 92–96. https://doi.org/10.1145/3511528.3511535
22. *Kulyabov D.S., Korolkova A.V.* Computer algebra in JULIA // Program. Comput. Software. 2021. Vol. 47. P. 133–138.
23. *Fedorov A.V., Masolova A.O., Korolkova A.V., Kulyabov D.S.* ApplicatioN of a numerical-analytical approach in the process of modeling differential equations in the Julia language // J. Phys.: Conf. Ser. 2020. V. 1694. № 1. P. 012026. https://doi.org/10.1088/1742-6596/1694/1/012026.
24. *Abotaleb M.S., Makarovskikh T.* Analysis of neural network and statistical models used for forecasting of a disease infection cases // Technol. Nanotechnol. 2021. P. 1–7.
25. *Tuluri F., Remata R., Walters W. L., Tchounwou P.B.* Application of machine learning to study the association between environmental factors and COVID-19 cases in Mississippi, USA // Mathematics. 2022. V. 10. № 6. P. 850. https://www.mdpi.com/2227-7390/10/6/850.
26. *Roman H.E., Croccolo F.* Spreading of infections on network models: Percolation clusters and random trees // Mathematics. 2021. V. 9. № 23. P. 3054. https://www.mdpi.com/2227-7390/9/23/3054.
27. *Romanyukha A.A.* Matematicheskie modeli v immunologii i epidemiologii infektsionnykh zabolevanii (Mathematical Models in Immunology and Epidemiology of Infectious Diseases), Moscow: Binom. Laboratoriya znanii, 2011.
28. *Kermack W.O., McKendrick A.G.* Contributions to the mathematical theory of epidemics // Proc. R. Soc. London, Ser. A. 1927. V. 115. P. 700–721.
29. *Strauss R.R., Bishnu S., Petersen M.R.* Comparing the performance of Julia on CPUs versus GPUs and Julia-MPI versus Fortran-MPI: A case study with MPAS-Ocean (Version 7.1), EGUsphere. 2023. V. 2023. P. 1–22.
30. *Rackauckas C., Nie Q.* Differentialequations.jl: A performant and feature-rich ecosystem for solving differential equations in Julia, J. Open Res. Software, 2017.
31. *Loman T.E., Ma Y., Ilin V., Gowda S., Korsbo N., Yewale N., Rackauckas C., Isaacson S.A.* Catalyst: Fast biochemical modeling with Julia, 2022. https://www.biorxiv.org/content/early/2022/08/02/2022.07.30.502135.
32. *Angevaare J., Feng, Z., Deardon R.* Pathogen.jl: Infectious disease transmission network modeling with Julia // J. Stat. Software. 2022. V. 104. № 4. P. 1–30. https://www.jstatsoft.org/index.php/jss/article/view/v104i04.
33. *Apreutesei A.M.Yu., Korolkova A.V., Kulyabov D.S.* Capabilities of hybrid modeling of systems with control in Modelica and Julia languages // Mater. XXIII Mezhdunar. Nauchn. Konf. “Raspredelennye komp’yuternye i telekommunikatsionnye seti: upravlenie, vychislenie, svyaz” (Proc. 23rd Int. Sci. Conf. Distributed Computer and Communication Networks: Control, Computation, Communications). 2020. P. 433–440.
34. *Apreutesey A.M.Y., Korolkova A.V., Kulyabov D.S.* Hybrid modeling of the red algorithm in the Julia language // J. Phys.: Conf. Ser. 7. 2020. P. 012025.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 45–50*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6

СИМВОЛЬНО-ЧИСЛЕННАЯ РЕАЛИЗАЦИЯ МОДЕЛИ АДИАБАТИЧЕСКИХ ВОЛНОВОДНЫХ МОД ДЛЯ ДВУМЕРНЫХ НЕРЕГУЛЯРНЫХ ВОЛНОВОДОВ

© 2024 г. Д. В. Диваков[a], *, А. А. Тютюнник[a], **, Д. А. Стариков[a], ***

[a] *Российский университет дружбы народов*
117198 Москва, ул. Миклухо-Маклая, д. 6, Россия

**E-mail: divakov_dv@pfur.ru*
***E-mail: tyutyunnik_aa@pfur.ru*
****E-mail: starikov_da@pfur.ru*

Поступила в редакцию 30.07.2023
После доработки 10.09.2023
Принята к публикации 01.10.2023

В работе построено символьно-численное решение уравнений Максвелла, описывающее направляемые моды двумерного плавно-нерегулярного волновода в рамках нулевого приближения модели адиабатических волноводных мод. Система линейных алгебраических уравнений, получаемая в нулевом приближении модели адиабатических волноводных мод, решена символьно. Дисперсионное уравнение решено численно методом продолжения по параметру.

Ключевые слова: символьное решение линейных уравнений, символьное решение дифференциальных уравнений, адиабатические волноводные моды, направляемые моды, плавно-нерегулярный волновод

DOI: 10.31857/S0132347424020066 **EDN:** ROVQMP

1. ВВЕДЕНИЕ

В работе строятся символьно-численные решения уравнений Максвелла, описывающие волноводные моды двумерного плавно-нерегулярного волновода в рамках модели адиабатических волноводных мод (АВМ) [1, 2, 3]. В основе модели АВМ лежит приближение коротких волн [4], адаптированное для волноводного распространения, которое представляет собой асимптотический ряд по малой величине, представляющей собой обратную частоту. Асимптотические методы удобны тем, что нулевое приближение отыскивается, как правило, в символьном виде. Учитывая асимптотический характер решения, в модели АВМ удается получить ряд промежуточных результатов в символьном виде – поэтому компьютерная алгебра используется в качестве основного инструмента исследования.

В нулевом приближении асимптотического разложения модели АВМ уравнения Максвелла символьно редуцируются к системе четырех обыкновенных дифференциальных уравнений первого порядка и двум дополнительным соотношениям. Систему дифференциальных уравнений удается решить символьно [5] в каждом слое многослойного волновода. Используя символьные представления решений формулируется система граничных уравнений в виде однородной системы линейных алгебраических уравнений (СЛАУ) с символьной матрицей коэффициентов. Однородная СЛАУ с символьной матрицей коэффициентов решается символьно в настоящей работе. Условие разрешимости однородной СЛАУ формулируется в виде нелинейного уравнения, которое решается в работе численно методом продолжения по параметру.

2. МЕТОДЫ

2.1. Модель адиабатических волноводных мод в нулевом приближении

В работе [5] получен нулевой вклад в адиабатическое приближение волноводного решения уравнений Максвелла вида:

$$\begin{Bmatrix} \vec{E}(x,z,t) \\ \vec{H}(x,z,t) \end{Bmatrix} = \begin{Bmatrix} \vec{E}_0(x;z) \\ \vec{H}_0(x;z) \end{Bmatrix} = \exp\{i\omega t - ik_0\varphi(z)\}, \quad (1)$$

причем

$$\begin{cases} \varepsilon\dfrac{\partial E_0^y}{\partial x} = -ik_0\varepsilon\mu H_0^z, \\ \varepsilon\dfrac{\partial E_0^z}{\partial x} = ik_0\left(\varepsilon\mu - \left(\varphi'(z)\right)^2\right)H_0^y, \\ \mu\dfrac{\partial H_0^y}{\partial x} = ik_0\varepsilon\mu E_0^z, \\ \mu\dfrac{\partial H_0^z}{\partial x} = -ik_0\left(\varepsilon\mu - \left(\varphi'(z)\right)^2\right)E_0^y, \end{cases} \quad (2)$$

$$E_0^x = \frac{1}{\varepsilon}\varphi'(z)H_0^y, H_0^x = -\frac{1}{\mu}\varphi'(z)E_0^y. \quad (3)$$

Систему уравнений (2)–(3) следует дополнить условиями сопряжения электромагнитного поля на границах раздела сред [6] для рассматриваемого многослойного волновода: на границах раздела диэлектрических сред выполняются граничные условия сопряжения полей

$$\left[n \times \vec{E}\right]_{x=0,h(z)} = \vec{0}, \quad \left[n \times \vec{H}\right]_{x=0,h(z)} = \vec{0}, \quad (4)$$

где через $[\vec{f}]_{x=0,h(z)}$ обозначен скачок векторной величины $\vec{f}$ на границах $x = 0, h(z)$. Кроме того, выполняются асимптотические граничные условия на бесконечности [6]:

$$\left\|\vec{E}\right\| \xrightarrow[x\to\pm\infty]{} 0, \left\|\vec{H}\right\| \xrightarrow[x\to\pm\infty]{} 0. \quad (5)$$

2.2. Геометрия рассматриваемой структуры

Рассматривается трехслойный плавно-нерегулярный волновод, геометрия которого представлена на рис. 1.

Параметры волновода следующие: $\mu_c = \mu_f = \mu_s = 1$, $\varepsilon_c = 1$, $\varepsilon_f = 1.565^2$, $\varepsilon_s = 1.47^2$, толщины определены как $h_1 = 2\lambda$, $h_2 = 3\lambda$, а $L = 100\lambda$, где λ – длина волны, $\lambda = 0.55$ [мкм]. Переменная толщина $h(z)$ определена следующим образом:

$$h(z) = 2(h_1 - h_2)\left(\frac{z}{L}\right)^3 - 3(h_1 - h_2)\left(\frac{z}{L}\right)^2 + h_1, \quad (6)$$

причем для $h'(z)$ выполняется $|h'(z)| \ll 1$, то есть $h'(z)$ является малым параметром при каждом фиксированном z.

В работе вычисляются адиабатические волноводные моды для описанной выше структуры в символьно-численном виде.

2.3. Символьный метод решения задачи

В работе система (2) решается в символьном виде. Коэффициенты системы (2) заданы в символьном виде, причем ε, μ для рассматриваемого многослойного волновода есть кусочно-постоянные функции – известные константы для каждого из слоев. В каждом слое систему (2) решаем символьно в системе компьютерной алгебры **Maple** с помощью функции **dsolve** [7] и получаем разложение решения по фундаментальной системе решений (ФСР) с неопределенными коэффициентами. Решения в полубесконечных слоях удовлетворяют условиям (5), поэтому константы, стоящие перед нарастающими на бесконечности функциями ФСР, будут определены и равны нулю.

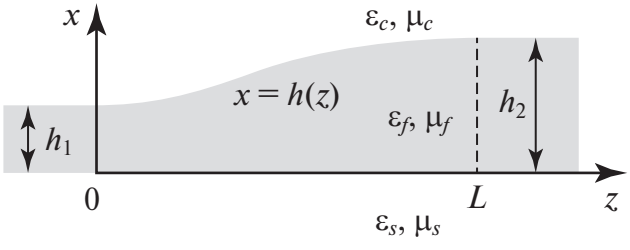


Рис. 1. Геометрия двумерного плавно-нерегулярного волноводного перехода между двумя регулярными волноводами.

Условия непрерывности (4) записываем символьно в системе компьютерной алгебры **Maple** [7] и получаем систему граничных уравнений вида

$$M\left(z, \gamma(z)\right)\vec{C}(z) = \vec{0}, \quad (7)$$

где $\gamma(z) = \varphi'(z)$, вектор $\vec{C}(z)$ определяет константы разложения решения по ФСР в каждом слое при фиксированном z. Условие разрешимости системы (7) есть равенство нулю определителя системы

$$\det M\left(z, \gamma(z)\right) = 0. \quad (8)$$

Система линейных алгебраических уравнений (7) с символьной матрицей коэффициентов решается в работе символьно методом из раздела 3.3 работы [8].

Символьное выражение детерминанта определяется с помощью функции **Determinant** пакета **LinearAlgebra**. Уравнение (8) характеризуется вещественными корнями при $z \le 0$ и при $z \ge L$, поэтому корни для $z = 0$ отыскиваются с помощью встроенной в **Maple** функции **RootOf** [7]. В интервале $0 < z < L$ корни уравнения (8) могут быть комплексными, поэтому для их отыскания в системе компьютерной алгебры использован метод продолжения по параметру, описанный ниже.

Описанный метод решения нелинейного уравнения (8), а также символьное решение системы (7) реализованы в системе компьютерной алгебры **Maple** [7].

2.4. Метод продолжения по параметру

Рассмотрим уравнение $F(x,y) = 0$ относительно искомой величины y, которое содержит параметр x. Пусть это уравнение имеет при $a \le x \le b$ решение $y(x)$ и при $x = a$ такое решение $y(a) = y_a$ известно, то есть $F(a,y)_a = 0$. Тогда в пространстве (x,y) точка y, соответствующая решениям рассматриваемого уравнения, описывает непрерывную кривую κ, проходящую через точки (a, y_a) и (b, y_b), где $F(b, y_b) = 0$.

Идея метода продолжения по параметру [9] состоит в построении решения отталкиваясь от (a, y_a) и двигаясь вдоль кривой к.

Дифференцирование рассматриваемого уравнения по параметру приводит к дифференциальному уравнению

$$\frac{dy}{dx} = -\frac{\partial F / \partial x}{\partial F / \partial y}, \tag{9}$$

которое вместе с условием

$$y(a) = y_a \tag{10}$$

образует задачу Коши.

В работе дифференциальное уравнение (9) формулируется в системе компьютерной алгебры с помощью встроенной в **Maple** функции **diff**. Начальное условие (10) формулируется численно в системе компьютерной алгебры с помощью непосредственного решения уравнения встроенной функцией **RootOf**.

Сформулированная таким образом задача Коши решается численно в системе компьютерной алгебры **Maple** с помощью функции **dsolve** методом **rkf45** с разными **abserr**, **relerr**.

3. РЕЗУЛЬТАТЫ

3.1. Символьное исследование системы

Система линейных алгебраических уравнений (7) *с символьной матрицей* коэффициентов исследуется символьно в системе компьютерной алгебры. После перестановки уравнений и искомых величин матрица системы (7) и вектор ее правых частей преобразуются к блочному виду

$$M = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix}, \vec{C} = \begin{pmatrix} \vec{C}_1 \\ \vec{C}_2 \end{pmatrix}. \tag{11}$$

и в результате система (7) разбиваеся на две независимых системы $M_1\vec{C}_1 = \vec{0}$ и $M_2\vec{C}_2 = \vec{0}$.

Матрица M_1 имеет следующую структуру

$$M_1 = \begin{pmatrix} m^1_{1,1} & m^1_{1,2} & m^1_{1,3} & 0 \\ 0 & m^1_{2,2} & m^1_{2,3} & m^1_{2,4} \\ 0 & m^1_{3,2} & m^1_{3,3} & m^1_{3,4} \\ m^1_{4,1} & m^1_{4,2} & m^1_{4,3} & 0 \end{pmatrix}, \tag{12}$$

элементы матрицы определены следующим образом

$$m^1_{1,1} = ih'\eta_c\varphi' - \eta_c^2, \tag{13}$$

$$m^1_{1,2} = ih'\eta_f\varphi' e^{\eta_f k_0 h} - \eta_f^2 e^{\eta_f k_0 h}, \tag{14}$$

$$m^1_{1,3} = -ih'\eta_f\varphi' e^{-\eta_f k_0 h} + \eta_f^2 e^{-\eta_f k_0 h}, \tag{15}$$

$$m^1_{2,2} = i\eta_f\varepsilon_f, \tag{16}$$

$$m^1_{2,3} = -i\eta_f\varepsilon_f, \tag{17}$$

$$m^1_{2,4} = -i\eta_s\varepsilon_s, \tag{18}$$

$$m^1_{3,2} = -\eta_f^2, \tag{19}$$

$$m^1_{3,3} = -\eta_f^2, \tag{20}$$

$$m^1_{3,4} = \eta_s^2, \tag{21}$$

$$m^1_{4,1} = -i\eta_c\varepsilon_c, \tag{22}$$

$$m^1_{4,2} = -i\eta_f\varepsilon_f e^{\eta_f k_0 h}, \tag{23}$$

$$m^1_{4,3} = i\eta_f\varepsilon_f e^{-\eta_f k_0 h}. \tag{24}$$

Матрица M_2 определена ниже

$$M_2 = \begin{pmatrix} 0 & m^2_{1,2} & m^2_{1,3} & m^2_{1,4} \\ 0 & m^2_{2,2} & m^2_{2,3} & m^2_{2,4} \\ m^2_{3,1} & m^2_{3,2} & m^2_{3,3} & 0 \\ m^2_{4,1} & m^2_{4,2} & m^2_{4,3} & 0 \end{pmatrix}, \tag{25}$$

элементы матрицы определены следующим образом

$$m^2_{1,2} = -\eta_f^2, \tag{26}$$

$$m^2_{1,3} = -\eta_f^2, \tag{27}$$

$$m^2_{1,4} = \eta_s^2, \tag{28}$$

$$m^2_{2,2} = i\eta_f\mu_f, \tag{29}$$

$$m^2_{2,3} = -i\eta_f\mu_f, \tag{30}$$

$$m^2_{2,4} = i\eta_s\mu_s, \tag{31}$$

$$m^2_{3,1} = ih'\eta_c\varphi' - \eta_c^2, \tag{32}$$

$$m^2_{3,2} = -ih'\eta_f\varphi' e^{-\eta_f k_0 h} + \eta_f^2 e^{-\eta_f k_0 h}, \tag{33}$$

$$m^2_{3,3} = ih'\eta_f\varphi' e^{\eta_f k_0 h} + \eta_f^2 e^{\eta_f k_0 h}, \tag{34}$$

$$m^2_{4,1} = i\eta_c\mu_c, \tag{35}$$

$$m^2_{4,2} = -i\eta_f\mu_f e^{-\eta_f k_0 h}, \tag{36}$$

$$m^2_{4,3} = i\eta_f\mu_f e^{\eta_f k_0 h}, \tag{37}$$

где $\eta_\alpha = \sqrt{\varphi'^2 - \varepsilon_\alpha\mu_\alpha}$, $\alpha = c, f, s$. Решение подсистемы $M_1\vec{C}_1 = \vec{0}$ получаем символьно, используя метод из раздела 3.3 работы [8]:

$$C_{1,1} = \frac{\eta_f^2\eta_s\left(\eta_s\varepsilon_f - \eta_f\varepsilon_s\right)\left(i\eta_f + h'\varphi'\right)e^{-\eta_f k_0 h}}{\eta_c\left(ih'\varphi' - \eta_c\right)} + \\ + \frac{\eta_f^2\eta_s\left(\eta_s\varepsilon_f + \eta_f\varepsilon_s\right)\left(i\eta_f - h'\varphi'\right)e^{\eta_f k_0 h}}{\eta_c\left(ih'\varphi' - \eta_c\right)}, \quad (38)$$

$$C_{1,2} = -i\eta_f\eta_s\left(\eta_s\varepsilon_f + \eta_f\varepsilon_s\right), \quad (39)$$

$$C_{1,3} = -i\eta_f\eta_s\left(\eta_s\varepsilon_f - \eta_f\varepsilon_s\right), \quad (40)$$

$$C_{1,4} = -2i\eta_f^3\varepsilon_f. \quad (41)$$

Решение подсистемы $M_2\vec{C}_2 = \vec{0}$ получаем аналогичным образом:

$$C_{2,1} = -\frac{i\eta_f^2\eta_s\mu_f\left(\eta_f\mu_s - \eta_s\mu_f\right)e^{-\eta_f k_0 h}}{\eta_c\mu_c} - \\ - \frac{i\eta_f^2\eta_s\mu_f\left(\eta_f\mu_s + \eta_s\mu_f\right)e^{\eta_f k_0 h}}{\eta_c\mu_c}, \quad (42)$$

$$C_{2,2} = i\eta_f\eta_s\left(\eta_s\mu_f - \eta_f\mu_s\right), \quad (43)$$

$$C_{2,3} = i\eta_f\eta_s\left(\eta_s\mu_f + \eta_f\mu_s\right), \quad (44)$$

$$C_{2,4} = 2i\eta_f^3\mu_f. \quad (45)$$

На основе полученных символьных решений (38)–(45) определяем символьное представление компонент электромагнитного поля в каждом из трех слоев рассматриваемого волновода.

Амплитудная часть электромагнитного поля в покровном слое ($x > h(z)$) определяется следующим образом:

$$\vec{E}_0^c = \begin{pmatrix} -i\eta_c\varphi' C_{1,1} \\ i\eta_c\mu_c C_{2,1} \\ \eta_c^2 C_{1,1} \end{pmatrix} e^{-\eta_c k_0 (x-h)}, \quad (46)$$

$$\vec{H}_0^c = \begin{pmatrix} -i\eta_c\varphi' C_{2,1} \\ -i\eta_c\varepsilon_c C_{1,1} \\ \eta_c^2 C_{2,1} \end{pmatrix} e^{-\eta_c k_0 (x-h)}. \quad (47)$$

Амплитудная часть электромагнитного поля в волноводном слое ($0 < x < h(z)$) определяется следующим образом:

$$\vec{E}_0^f = \begin{pmatrix} i\eta_f\varphi' C_{1,2} \\ -i\eta_f\mu_f C_{2,3} \\ \eta_f^2 C_{1,2} \end{pmatrix} e^{\eta_f k_0 x} + \begin{pmatrix} -i\eta_f\varphi' C_{1,3} \\ i\eta_f\mu_f C_{2,2} \\ \eta_f^2 C_{1,3} \end{pmatrix} e^{-\eta_f k_0 x}, \quad (48)$$

$$\vec{H}_0^f = \begin{pmatrix} i\eta_f\varphi' C_{2,3} \\ i\eta_f\varepsilon_f C_{1,2} \\ \eta_f^2 C_{2,3} \end{pmatrix} e^{\eta_f k_0 x} + \begin{pmatrix} -i\eta_f\varphi' C_{2,2} \\ -i\eta_f\varepsilon_f C_{1,3} \\ \eta_f^2 C_{2,2} \end{pmatrix} e^{-\eta_f k_0 x}. \quad (49)$$

Амплитудная часть электромагнитного поля в подложке ($x < 0$) определяется следующим образом:

$$\vec{E}_0^s = \begin{pmatrix} i\eta_s\varphi' C_{1,4} \\ -i\eta_s\mu_s C_{2,4} \\ \eta_s^2 C_{1,4} \end{pmatrix} e^{\eta_s k_0 x}, \quad (50)$$

$$\vec{H}_0^s = \begin{pmatrix} i\eta_s\varphi' C_{2,4} \\ i\eta_s\varepsilon_s C_{1,4} \\ \eta_s^2 C_{2,4} \end{pmatrix} e^{\eta_s k_0 x}. \quad (51)$$

В приведенных символьных выражениях (46)–(51) только для величины φ' неизвестно символьное представление, так как она является решением нелинейного уравнения (8) и определяется численно.

3.2. Численное решение нелинейного уравнения

Используем для решения нелинейного уравнения (8) метод продолжения по параметру, описанный в разделе 1 настоящей работы. Вычисленные величины $\gamma_j(z)$, $j = 1..4$ приведены на рис. 2, 3. Приведем также невязки решений, полученных методом продолжения по параметру. Величины $\delta_j(z) = \left|\det M(z, \gamma_j(z))\right|, j = 1..4$ (невязки) приведены на рис. 4, 5.

4. ОБСУЖДЕНИЕ

В работе получены символьные представления направляемых мод плавно-нерегулярного волноводного перехода (рис. 1) между двумя регулярными

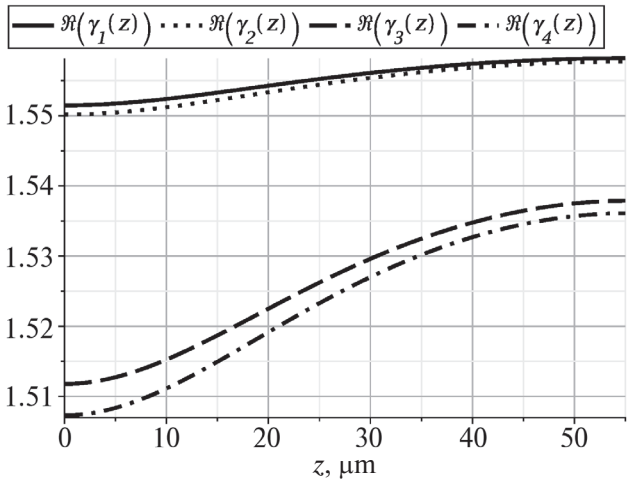


Рис. 2. Величины $\Re(\gamma_j(z))$, $j = 1..4$ для $z \in [0, L]$.

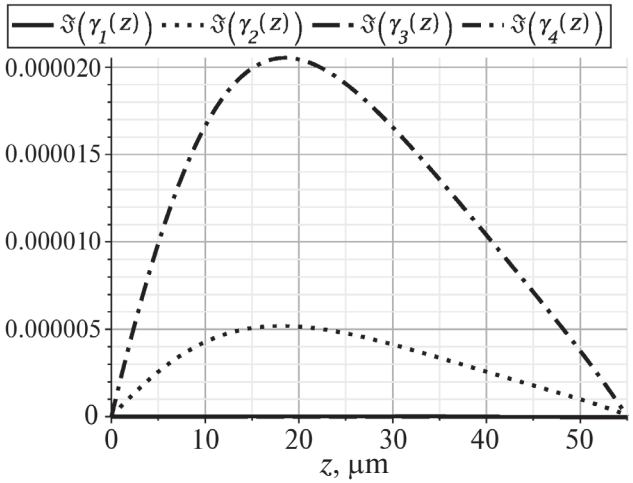


Рис. 3. Величины $\Im(\gamma_j(z))$, $j = 1..4$ для $z \in [0, L]$.

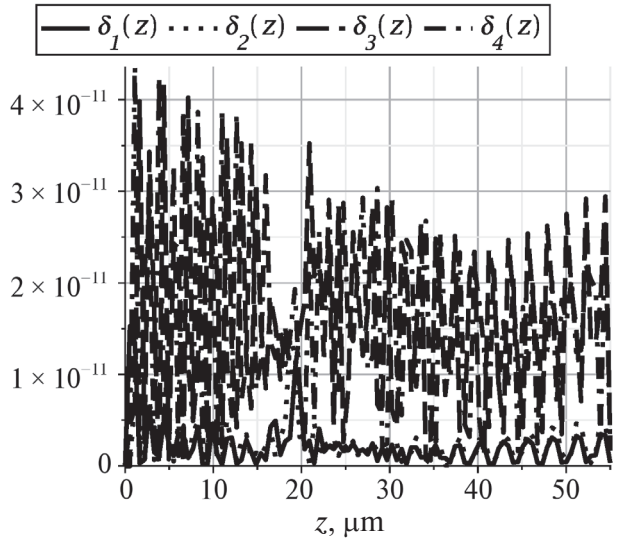


Рис. 4. Невязки $\delta_j(z)$, $j = 1..4$ для $z \in [0, L]$ при значениях **abserr** и **relerr**, равных 10^{-10}.

волноводами в рамках модели адиабатических волноводных мод. Символьные представления направляемых мод во всех слоях волновода достаточно компактны (46)–(51) и позволяют построить волноводные моды в символьно-численном виде, где численно отыскивается только $\gamma(z) = \varphi'(z)$. Кроме того, символьный вид направляемых мод модели АВМ в нулевом приближении асимптотического разложения (46)–(51) будет использован при построении первого приближения асимптотического метода.

Важно отметить, что полученные символьные выражения (46)–(51) не единственны. Каждая подсистема $M_1\vec{C}_1 = \vec{0}$ и $M_2\vec{C}_2 = \vec{0}$ имеет по 4 различных символьных представлений решения, как в работе [8]. Вектор решения $\vec{C}$, определенный в (11), будет иметь 16 различных символьных представлений.

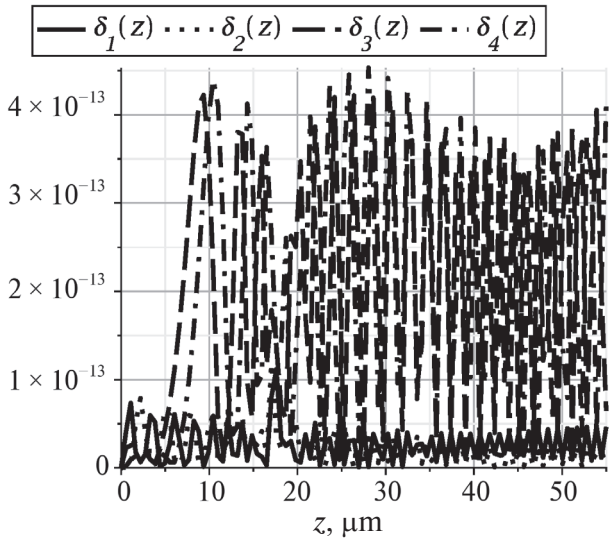


Рис. 5. Невязки $\delta_j(z)$, $j = 1..4$ для $z \in [0, L]$ при значениях **abserr** и **relerr**, равных 10^{-12}.

Кроме того, любая линейная комбинация этих 16 символьных представлений также будет решением однородной системы (7). Вопрос выбора наиболее удобного символьного представления решения остается открытым.

В рамках приближенного вычисления корней нелинейного уравнения вычислены функции $\gamma_j(z)$, описывающие переменные коэффициенты фазового замедления для j-й волноводной моды. В работе [10] приближенно вычислялись функции $\gamma_j(z)$ методом разложения по малому параметру $h'(z)$, отвечающему малому наклону криволинейной границы раздела. В работе [10] построены численно нулевое, первое и второе приближения по малому параметру и вычислены невязки. В обеих работах рассматривается одинаковая структура.

В настоящей работе вместо разложения по малому параметру использован метод продолжения по параметру, который дает невязку, не превышающую 4.5×10^{-11}, согласно рис. 4, и невязку, не превышающую 4.5×10^{-13} согласно рис. 5. В работе [10] получена меньшая точность порядка 10^{-8}. Другими словами, точность метода продолжения по параметру на несколько порядков выше, чем у метода малого параметра из [10], и меняется в зависимости от значений параметров **abserr**, **relerr** метода rkf45. В методе малого параметра из [10] решение представляется в виде разложения, коэффициенты которого известны в виде довольно громоздких символьных выражений. Обобщая вышесказанное, метод малого параметра менее удобен для решения нелинейного уравнения (8), чем использованный в настоящей работе метод продолжения по параметру.

5. ЗАКЛЮЧЕНИЕ

Модель адиабатических волноводных мод позволяет сформулировать задачу расчета направляемых мод в символьном виде и решить ее в символьно-численном виде.

Использование компьютерной алгебры позволило получить символьные выражения для адиабатических волноводных мод в нулевом приближении для волноводного перехода.

Описан метод продолжения по параметру и реализован в системе компьютерной алгебры для решения дисперсионных уравнений с комплексными корнями. Продемонстрированы эффективность и высокая точность метода.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Публикация выполнена при поддержке Программы стратегического академического лидерства РУДН, проект № 021934-0-000.

СПИСОК ЛИТЕРАТУРЫ

1. *Sevastianov L.A., Egorov A.A.* Theoretical analysis of the waveguide propagation of electromagnetic waves in dielectric smoothlyirregular integrated structures // Optics and Spectroscopy. 2008. V. 105. № 4. P. 576–584.
2. *Egorov A.A., Sevastianov L.A.* Structure of modes of a smoothly irregular integrated optical four-layer three-dimensional waveguide // Quantum Electronics. 2009. V. 39. № 6. P. 566–574.
3. *Egorov A.A., Lovetskiy K.P., Sevastianov A.L., Sevastianov L.A.* Simulation of guided modes (eigenmodes) and synthesis of a thin-film generalised waveguide Luneburg lens in the zero-order vector approximation // Quantum Electronics. 2010. V. 40. № 9. P. 830–836.
4. *Babich V.M., Buldyrev V.S.* Asimptotic Methods in Short-Wave Diffraction Problems. Method of Reference Problems, Moscow: Nauka, 1972.
5. *Divakov D.V., Sevastianov A.L.* The Implementation of the Symbolic-Numerical Method for Finding the Adiabatic Waveguide Modes of Integrated Optical Waveguides in CAS Maple // Lecture Notes in Computer Science. 2019. V. 11661. P. 107–121.
6. *Adams M.J.* An Introduction to Optical Waveguides. Wiley, New York (1981).
7. Mathematics-based software and services for education, engineering, and research https://www.maplesoft.com/
8. *Divakov D.V., Tyutyunnik A.A.* Symbolic investigation of the spectral characteristics of guided modes in smoothly irregular waveguides // Program. Comput. Software. 2022. V. 48. № 2. P. 80–89.
9. *Kuznetsov E.B., Shalashilin V.I.* Solution of differential-algebraic equations using the parameter continuation method // Differ. Uravn. 1999. V. 35. № 3. P. 379–387.
10. *Divakov D.V., Tyutyunnik A.A.* Symbolic-numerical modeling of adiabatic waveguide mode in a smooth waveguide transition // Comput. Math. Math. Phys. 2023. V. 63. № 1. P. 95–105.

SYMBOLIC-NUMERICAL IMPLEMENTATION OF THE MODEL OF ADIABATIC GUIDED MODES FOR TWO-DIMENSIONAL IRREGULAR WAVEGUIDES

© 2024 D. V. Divakov[a], A. A. Tyutyunnik[a], D. A. Starikov[a]

[a] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198 Russia*

In this work, a symbolic-numerical solution of Maxwell's equations is constructed, describing the guided modes of a two-dimensional smoothly irregular waveguide in the zeroth approximation of the model of adiabatic waveguide modes. The system of linear algebraic equations obtained in this approximation is solved symbolically. The dispersion relation is solved numerically using the parameter continuation method.

Keywords: symbolic solution of linear equations, symbolic solution of differential equations, adiabatic waveguide modes, guided modes, smoothly irregular waveguide

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 51–65*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.422.8.

РЕАЛИЗАЦИЯ АНАЛИТИЧЕСКОЙ ПРОЕКТИВНОЙ ГЕОМЕТРИИ ДЛЯ КОМПЬЮТЕРНОЙ ГРАФИКИ

© 2024 г. М. Н. Геворкян[a, *], А. В. Королькова[a, **], Д. С. Кулябов[a, b, *], Л. А. Севастьянов[a, b, ****]**

[a]*Российский университет дружбы народов, 117198 Москва, ул. Миклухо-Маклая, д. 6, Россия*

[b]*Объединенный институт ядерных исследований, 141980 Дубна, Московская обл., ул. Жолио-Кюри, д. 6, Россия*

**E-mail: gevorkyan-mn@rudn.ru*

***E-mail: korolkova-av@rudn.ru*

****E-mail: kulyabov-ds@rudn.ru*

*****E-mail: sevastianov-la@rudn.ru*

Поступила в редакцию 09.08.2023
После доработки 10.09.2023
Принята к публикации 01.10.2023

В своих исследованиях авторы активно используют разные разделы геометрии. Для геометрических построений используются подходы и системы компьютерной алгебры. В данный момент нас заинтересовала такая область, как компьютерная геометрия, и более узко, реализация машинной графики. Стандартом де-факто в современной компьютерной графике стало использование проективного пространства и однородных координат, то есть задача фактически сводится к применению аналитической проективной геометрии. Авторам не удалось подобрать систему компьютерной алгебры, которая могла бы реализовать проективную геометрию во всем объеме. Поэтому было принято решение реализовать применение компьютерной алгебры частично, для визуализации алгебраических ссотношений. Для этого предлагается использовать систему Asymptote.

Ключевые слова: проективная геометрия, система Asymptote, координаты Плюккера, собственные и несобственные точки, прямые и плоскости

DOI: 10.31857/S0132347424020089 **EDN:** ROPXHV

1. ВВЕДЕНИЕ

Проективная геометрия находит применение в разных областях компьютерных наук: в компьютерной графике, робототехнике, машинном зрении. Математическая теория служит основой для создания алгоритмов и их программной реализации. Авторы считают, что данный математический раздел имеет большие прикладные перспективы.

В проективной геометрии непропорционально большую роль играет визуализация геометрических структур. Авторам не удалось найти полной системы компьютерной алгебры, специализирующейся на методах проективной геометрии, что несомненно связано с ее спецификой. Ранее мы использовали методы проективной геометрии для обоснования применения гиперкомплексных чисел для описания движений в разных вариантах физических пространств [1–4]. В качестве следующего шага мы использовали подход геометрической алгебры (с использованием систем компьютерной алгебры) для исследования тех же объектов под другим углом зрения [5–7]. На следующем шаге мы решили вернуться к прямому применению теоретических основ к исследованиям. В качестве первого этапа данной научной программы предлагается частичная компьютеризация исследований. Алгебраические формулы реализованы в виде структур данных и наборов функций, связанных с геометрическими структурами. Затем с помощью языка Asymptote сделана программная реализация для визуализации полученных алгебраических структур. В данном случае к структурам относятся точки, прямые и плоскости.

Язык Asymptote имеет структуру системы компьютерной алгебры, но вместо символьного представления алгебраических объектов в результате работы Asymptote получается их визуальное представление.

Система Asymptote [8–12] — специализированный высокоуровневый язык программирования для создания векторных изображений научной направленности (геометрические чертежи, схемы, диаграммы и т.д.). Язык использует C-подобный синтаксис с некоторым числом дополнений и улучшений[1], что по замыслу разработчиков делает его более

[1] Сам язык в своем синтаксисе скорее ориентируется на C++ [13]. Например, переменные в Asymptote можно объявлять в любом месте программы и инициализировать сразу при объявлении, а функции могут принимать параметры по умолчанию.

удобным для освоения и использованием по сравнению с MetaPost [14] и PostScript.

Для наших целей крайне важно, что Asymptote поддерживает создание полноценных трехмерных изображений благодаря использованию OpenGL с автоматическим расчетом взаимного расположения геометрических тел, источников света и тени, что позволяет изображать пересечение поверхностей и кривых. Однако программный интерфейс для построения трехмерных изображений гораздо менее проработан и скупее документирован по сравнению с возможностями двумерной графики.

В данной статье мы не будем излагать основ языка Asymptote. Заинтересованный читатель может найти описание всех его возможностей в официальном руководстве на сайте [15], базовые примеры приведены в обучающих заметках [11], также множество примеров с комментариями на русском языке можно посмотреть в источниках [16–18]. Следует отметить, что большая часть примеров относится к двумерным изображениям. В статье акцент сделан на описании реализации посредством Asymptote структур данных, таких как проективные точки, прямые и плоскости для трехмерных графиков, приведены примеры исходного кода с пояснениями.

1.1. Структура статьи

В начале статьи приведена реализация структур данных, описывающих проективные точки (раздел 2), прямые (раздел 3) и плоскости (раздел 4). Изложение сопровождается примерами исходного кода с пояснением всех специфических конструкций, ключевых слов и функций языка Asymptote. В разделе 5 приводятся несколько примеров использования созданных структур, а также описываются некоторые технические тонкости создания именно трехмерных изображений, так как в официальном руководстве данный аспект освещен довольно скупо.

1.2. Обозначения и соглашения

Напомним основные используемые обозначения проективной геометрии. Формулы приведены в однородных координатах, обозначения взяты из монографии [19]:

• $\{\mathbf{v} \mid \mathbf{p} \times \mathbf{v}\}$ — прямая, проходящая через точку P по направлению $\mathbf{v}$;

• $\{\mathbf{p}_2 - \mathbf{p}_1 \mid \mathbf{p}_1 \times \mathbf{p}_2\}$ — прямая, проходящая через две точки P_1 и P_2;

• $\{\mathbf{p} \mid \mathbf{0}\}$ — прямая, проходящая через начало координат и точку P;

• $\{w_1\mathbf{p}_2 - w_2\mathbf{p}_1 \mid \mathbf{p}_1 \times \mathbf{p}_2\}$ — прямая, проходящая через две точки $\vec{\mathbf{p}}_1 = (\mathbf{p}_1 \mid w_1)$ и $\vec{\mathbf{p}}_2 = (\mathbf{p}_2 \mid w_2)$;

• $\{\mathbf{n}_1 \times \mathbf{n}_2 \mid d_1\mathbf{n}_2 - d_2\mathbf{n}_1\}$ — прямая линия пересечения двух плоскостей $[\mathbf{n}_1 \mid d_1]$ и $[\mathbf{n}_2 \mid d_2]$;

• $(m \times n + dv \mid -(n, v))$ — точка пересечения плоскости $[n \mid d]$ и прямой $\{v \mid m\}$;

• $[m_1 \times m_2 \mid (v_2, m_1)]$ — точка пересечения двух прямых $\{v_1 \mid m_1\}$ и $\{v_2 \mid m_2\}$;

• $(d_1 n_3 \times n_2 + d_2 n_1 \times n_3 + d_3 n_2 \times n_1 \mid (n_1, n_2, n_3))$ — точка пересечения трех плоскостей $[n_1 \mid d_1]$, $[n_2 \mid d_2]$ и $[n_3 \mid d_3]$;

• $(v \times m \mid (v, v))$ — точка, ближайшая к началу координат на прямой $\{v \mid m\}$;

• $(-dn \mid (n, n))$ — точка, ближайшая к началу координат на плоскости $[n \mid d]$;

• $[v \times u \mid -(u, m)]$ — плоскость, содержащая прямую $\{v \mid m\}$ и направление u;

• $[v \times p + m \mid -(p, m)]$ — плоскость, содержащая прямую $\{v \mid m\}$ и точку $(p \mid 1)$;

• $[m \mid 0]$ — плоскость, содержащая прямую $\{v \mid m\}$ и начало координат;

• $[v \times p + wm \mid -(p, m)]$ — плоскость, содержащая прямую $\{v \mid m\}$ и точку $(p \mid w)$;

• $[v \times u \mid (u, v, p)]$ — плоскость, содержащая точку $(p \mid 1)$ и направления v и u;

• $[m \times v \mid (m, m)]$ — плоскость с прямой $\{v \mid m\}$, наиболее отдаленная от O;

• $[-wp \mid (p, p)]$ — плоскость с точкой $(p \mid w)$, наиболее отдаленная от O;

• $\dfrac{(v_1, m_2) + (v_2, m_1)}{v_1 \times v_2}$ — расстояние между прямыми $\{v_1 \mid m_1\}$ и $\{v_2 \mid m_2\}$;

• $\dfrac{v \times p + m}{v}$ — расстояние между прямой $\{v \mid m\}$ и точкой $(p \mid 1)$;

• $m \,/\, v$ — расстояние от прямой $\{v \mid m\}$ до начала координат;

• $\dfrac{(n, p) + d}{n}$ — расстояние от плоскости $[n \mid d]$ до точки $(p \mid 1)$;

• $d \,/\, n$ — расстояние от плоскости $[n \mid d]$ до начала координат.

2. СОЗДАНИЕ СТРУКТУР НА ПРИМЕРЕ ОДНОРОДНЫХ КООРДИНАТ ТОЧКИ

Для создания пользовательских структур в языке Asymptote используется синтаксис, во многом похожий на соответствующий синтаксис языка Си, однако имеющий некоторую специфику, которую мы рассмотрим на примере структуры для однородных координат.

Напомним, что однородные координаты представляют собой систему координат, используемую в проективной геометрии и обладающую тем свойством, что определяемый ими объект не меняется при умножении всех координат на одно и то же ненулевое число. Из-за этого количество координат, необходимое для представления точек, всегда на одну больше, чем размерность пространства, в котором эти координаты используются.

Язык Asymptote имеет встроенный тип triple, который используется для задания координат точек в трехмерном пространстве. Структура данных для однородных координат отсутствует, по крайней мере недоступна пользователям, поэтому следует ее создать.

Назовем структуру Vec4, по аналогии с соответствующим типом данных из GLSL (OpenGL Shading Language, Graphics Library Shader Language):

```
struct Vec4 {
  real x; real y; real z; real w;
  pair xy; triple xyz;
  real[] xyzw;
}
```

Структура данных должна содержать всего 4 числа типа real, однако для удобства использования кроме полей x, y, z, w, соответствующих однородным координатам $x : y : z : w$, мы задаем поля, позволяющие получить только координаты xy в виде типа данных pair, трехмерные декартовы координаты xyz в виде типа данных triple и все координаты в виде массива $xyzw$. Кроме удобства при манипуляциях с указанными объектами такой подход также повторяет программный интерфейс языка GLSL.

После определения структуры Vec4 Asymptote автоматически создает одноименную функцию, обязательными аргументами которой являются все поля, перечисленные при определении структуры. Данная функция эквивалентна конструктору по умолчанию в терминах языка C++ или Java.

В нашем случае придется перечислить все семь полей, хотя они частично дублируют друг друга. Данная проблема решается заданием специализированных операторов инициализации (конструкторов), для чего следует перегрузить специальный оператор init. Внутри этого оператора можно использовать ключевое слово this, которое позволяет сослаться на экземпляр структуры.

Для нашего примера создадим три специализированных конструктора:

• Конструктор Vec4 из набора чисел:

```
void operator init(real x, real y,
↪  real z, real w) {
this.x = x;
this.y = y;
this.z = z;

  this.w = w;
  this.xy = (x, y);
  this.xyz = (x, y, z);
  this.xyzw = new real[] {this.x,
  ↪  this.y, this.z, this.w};
}
```

• Конструктор Vec4 из декартовых координат точки и отдельного скаляра w:

```
void operator init(triple p, real w)
↪  {
  this.x = p.x;
  this.y = p.y;
  this.z = p.z;
  this.w = w;
  this.xy = (p.x, p.y);
  this.xyz = (p.x, p.y, p.z);
  this.xyzw = new real[] {this.x,
  ↪  this.y, this.z, this.w};
}
```

• Конструктор Vec4 из массива:

```
void operator init(real[] v) {
  this.x = v[0];
  this.y = v[1];
  this.z = v[2];
  this.w = v[3];
  this.xy = (v[0], v[1]);
  this.xyz = (v[0], v[1], v[2]);
  this.xyzw = v[0:5];
}
```

Все перечисленные операторы следует размещать внутри пространства имен структуры (т.е. между открывающей и закрывающей фигурными скобками структуры Vec4).

Для удобства манипулирования данными можно также перегрузить оператор cast, который ответственен за неявное преобразование типов данных:

• Преобразование типа Vec4 в массив:

```
real[] operator cast(Vec4 v) {
  return v.xyzw;
}
```

• Преобразование типа Vec4 в трехмерные декартовы координаты:

```
triple operator cast(Vec4 v) {
  return v.xyz / v.w;
}
```

• Преобразование массива в тип Vec4:

```
Vec4 operator cast(real[] v) {
  return Vec4(v);
}
```

При преобразовании данных в тип triple осуществляется дополнительное деление на весовую координату w. В случае несобственной (идеальной) точки данное преобразование сработает некорректно, так как у идеальной точки координата $w=0$ и произойдет деление на ноль. Однако проверку на равенство нулю координаты w всегда можно произвести отдельно.

3. ПРОЕКТИВНЫЕ ПРЯМЫЕ В ВИДЕ СТРУКТУРЫ LINE3D

3.1. Структура и конструкторы

Прямая в пространстве полностью определяется координатами Плюккера, которые задаются направляющим вектором v и вектором момента m. Для вычислений удобно, чтобы вектор v был единичными. На основе векторов v и m можно вычислить любые другие характеристики прямой, поэтому в качестве полей структуры достаточно использовать только эти два вектора. Однако для вычислений часто необходимо знать некоторую точку P прямой, поэтому удобно также включить ее в поля структуры. Также для большей общности можно допустить хранение неединичного вектора v. В итоге структура для задания проективной прямой будет иметь следующий вид:

```
struct Line3D {
  // Направляющий вектор и момент:
  triple V, m;
  // Единичный направляющий вектор:
  triple v;
  // Произвольная точка прямой:
  triple P;
}
```

Для полноценного использования созданной структуры в ее области имен необходимо задать несколько дополнительных операторов инициализации. Так, прямая, проходящая через две собственные точки, заданные в декартовых координатах, вычисляется по формуле:

$$\mathbf{m} = \mathbf{p}_1 \times \mathbf{p}_2. \qquad (1)$$

Соответствующий конструктор будет выглядеть следующим образом:

```
void operator init (triple P2, triple
↪  P1) {
  this.V = P2 - P1;
  this.v = unit(this.V);
  this.m = cross(P1, P2) /
  ↪  length(this.V);
  this.P = P1;
}
```

Следует обратить внимание, что для вычисления единичного вектора v мы использовали встроенную в Asymptote функцию unit, для вычисления векторного произведения — функцию cross, а для вычисления нормы вектора — функцию length. Обратите внимание, что при вычислении m используется функция (1) в комбинации с соотношением

$$d_{OO_\perp} = OO_\perp =$$

$$= \frac{v \times m}{v} = \frac{vm\sin\frac{\pi}{2}}{v^2} = \frac{m}{v},$$

т.е. она делится на исходную длину вектора v, вычисленного как $\mathbf{p}_2 - \mathbf{p}_1$. В этом случае длина m имеет геометрический смысл расстояния от начала координат до прямой.

Следующий конструктор инициализирует структуру по заданным векторам $\mathbf{v}$ и $\mathbf{m}$:

```
void operator init (triple keyword V,
↪  triple keyword m) {
  this.V = V;
  this.v = unit(this.V);
  this.m = m / length(this.V);
  this.P = cross(this.v, this.m);
}
```

Вектор $\mathbf{v}$ может быть произвольной длины, его исходное значение записывается в поле V структуры, а его нормированное значение — в поле v. Здесь вектор $\mathbf{m}$ нормализован путем деления на собственную длину ненормированного касательного вектора $\|\mathbf{v}\|$.

Инструкция keyword указывает, что данный оператор инициализации можно вызывать только явно указав все аргументы, т.е. следующим образом:

```
triple V = (1,1,1); triple m = (1, 1,
↪  1);
Line3D L = Line3D(V=V, m=m);
```

Если осуществить вызов через Line3D L = Line(V, m), то будет вызван предыдущий конструктор по двум точкам, заданным в декартовых координатах.

В реализации конструктора, который инициализирует прямую, проходящую через две точки, заданные в проективных координатах, используется следующая формула:

$$\{w_1\mathbf{p}_2 - w_2\mathbf{p}_1 \mid \mathbf{p}_1 \times \mathbf{p}_2\}. \qquad (2)$$

Соответствующий конструктор будет выглядеть следующим образом:

```
void operator init (Vec4 P2, Vec4 P1) {
  this.V = P1.w * P2.xyz - P2.w *
  ↪  P1.xyz;
  this.v = this.V / length(this.V);
  this.m = cross(P1.xyz, P2.xyz)  /
  ↪  length(this.V);
  this.P = P1.xyz / P1.w;
}
```

Мы неявно подразумеваем, что по крайней мере P_1 — собственная точка, т.е. ее координата w не равна нулю. Подчеркнем, что формула (3) — универсальна и работает для всех возможных вариантов точек. Например, если $\vec{\mathbf{p}}_1 = (x, y, z, w)$ — собственная точка, а $\vec{\mathbf{p}}_2 = (\mathbf{v} \mid 0) = (v_x, v_y, v_z, 0)$ — несобственная точка, то координаты Плюккера прямой вычисляются как

$$\vec{\mathbf{l}} = \{w\mathbf{v} \mid \mathbf{p} \times \mathbf{v}\}.$$

Произведя нормализацию путем деления на $w|v|$, получим формулу:

$$\{\mathbf{v} \mid \mathbf{m}\} = \{v_x, v_y, v_z \mid m_x, m_y, m_z\} = \{\mathbf{v} \mid \mathbf{p} \times \mathbf{v}\}. \quad (3)$$

Формуле (3) соответствует следующий конструктор (прямая, проходящая через точку по направляющему вектору):

```
void operator init (Vec4 P, triple V) {
  this.V = V;
  this.v = this.V / length(this.V);
  this.m = cross(P.xyz, this.V) /
  ↪  length(this.V);
  this.P = P.xyz;
}
```

Если же обе точки $\vec{\mathbf{p}}_1 = (\mathbf{v}_1 \mid 0)$ и $\vec{\mathbf{p}}_2 = (\mathbf{v}_2 \mid 0)$ являются несобственными, то получаем несобственную прямую $\{\mathbf{0} \mid \mathbf{v}_1 \times \mathbf{v}_2\}$. Несобственные прямые могут встречаться при рассмотрении взаимного расположения плоскостей. Если плоскости параллельны, то они будут пересекаться как раз по несобственным прямым. Пример формулы (3) иллюстрирует преимущество проективного подхода, который позволяет одной формулой охватить сразу все возможные варианты.

Осталось создать еще два конструктора, которые реализуют формулу

$$\{\mathbf{v} \mid 0\}.$$

Соответствующие конструкторы имеют вид:

• Прямая, проходящая через начало координат и точку P (в декартовых координатах):

```
void operator init (Vec4 P) {
  this.V = P.xyz;
  this.v = unit(this.V);
  this.m = (0, 0, 0);
  this.P = P.xyz;
}
```

• Прямая, проходящая через начало координат и точку P:

```
void operator init (triple P) {
  this.V = P;
  this.v = unit(this.V);
  this.m = (0, 0, 0);
  this.P = P;
}
```

Для визуализации прямой необходимо знать хотя бы две ее точки. Для реализации параметрического уравнения прямой определим в пространстве имен структуры Line3D следующую функцию:

```
triple get_point(real t) {
  return this.P + this.v * t;
}
```

Удобно также иметь возможность вычислить сразу массив точек прямой, что крайне просто делается с помощью поэлементного цикла for:

```
triple[] get_points(real[] T) {
  triple[] res;
  for (real t : T) {
    res.push(get_point(t));
  }
  return res;
}
```

Данный вариант цикла for похож на аналогичный цикл из языка Python, позволяет перебирать элементы из массива и проводить с ними манипуляции.

3.2. Взаимное расположение прямых

В предыдущем пункте мы задали поля структуры, набор конструкторов и несколько методов. Теперь зададим несколько функций, аргументами которых будут экземпляры структуры Line3D. Данные функции разместим в том же файле, что и структура Line3D, но вне ее пространства имен (за закрывающей фигурной скобкой).

В первую очередь реализуем функцию вычисления *взаимного момента* двух прямых l_1 и l_2 по простой формуле

$$M = (\mathbf{v}_1, \mathbf{m}_2) + (\mathbf{v}_2, \mathbf{m}_1).$$

Реализация соответствующей функции имеет вид:

```
real reciprocal_moment(Line3D line01,
 ↪  Line3D line02) {
   return dot(line01.m, line02.v) +
    ↪  dot(line01.v, line02.m);
}
```

Здесь мы использовали встроенную функцию dot, которая вычисляет скалярное произведение двух векторов.

Несмотря на свою простоту, данная функция крайне полезна и позволяет определить взаимное положение двух прямых. Возможны три случая в зависимости от знака взаимного момента:

- если $M=0$, то прямые лежат в одной плоскости;
- если $M>0$, то переход от l_1 к l_2 осуществляется правым поворотом;
- если $M<0$, то переход от l_1 к l_2 получается левым поворотом.

Заметим, что делать проверку на равенство нулю следует с учетом погрешности представления вещественных чисел числами с плавающей точкой.

Далее определим функцию, реализующую формулу

$$d = d_2 - d_1 = \frac{(v_1, m_2) + (v_2, m_1)}{v_1 \times v_2}. \qquad (4)$$

Соотношение (4) позволяет вычислить длину d взаимного перпендикуляра между двумя скрещивающимися прямыми. Так как у нас уже есть функция вычисления взаимного момента, то вычисления длины перпендикуляра будет осуществляться также в одну строчку:

```
real reciprocal_perp_length(Line3D line01,
 ↪  Line3D line02) {
   return reciprocal_moment(line01, line02)
    ↪  / length(cross(line01.v, line02.v));
}
```

Предыдущую функцию можно было бы сделать более общей, добавив проверку скрещиваемости прямых, и в случае, если прямые лежат в одной плоскости, применить формулу

$$d_{QQ_\perp} = \frac{\|\mathbf{m} + \mathbf{v} \times \mathbf{q}\|}{\|\mathbf{v}\|} = \frac{\|(\mathbf{p} - \mathbf{q}) \times \mathbf{v}\|}{\|\mathbf{v}\|}$$

в виде

$$d = \frac{\mathbf{v} \times \mathbf{p}_2 + \mathbf{m}_1}{\mathbf{v}_1},$$

где $\vec{\mathbf{l}}_1 = \{\mathbf{v}_1 \,|\, \mathbf{m}_1\}$ — координаты Плюккера первой прямой, $\vec{\mathbf{p}}_2 = (\mathbf{p}_2 \,|\, 1)$ — однородные координаты произвольной точки второй прямой. Однако мы не стали усложнять функцию дополнительными внутренними проверками. Также реализуем формулу

$$[\mathbf{m}_1 \times \mathbf{m}_2 \,|\, (\mathbf{v}_2, \mathbf{m}_1)] = [\mathbf{m}_2 \times \mathbf{m}_1 \,|\, (\mathbf{v}_1, \mathbf{m}_2)]. \qquad (5)$$

При этом сделаем проверку на равенство нулю взаимного момента, так как в этом случае две прямые будут лежать в одной плоскости (будут компланарны):

```
// line-line-meet
Vec4 intersection(Line3D line01, Line3D
 ↪  line02) {
    real M = reciprocal_moment(line01,
     ↪  line02);
  assert( abs(M) < 1e-8, "Прямые не
   ↪  компланарны");
  return Vec4( cross(line01.m, line02.m),
   ↪  dot(line02.v, line01.m));
  // return Vec4( cross(line02.m,
   ↪  line01.m), dot(line01.v, line02.m));
}
```

Как указывалось выше, проверку на равенство нулю следует сделать с некоторой погрешностью. В нашей реализации мы допускаем довольно большую погрешность, так как основной целью является визуализация полученных объектов, где малые погрешности не критичны.

Для вычисления координат Плюккера прямой, содержащей взаимный перпендикуляр, воспользуемся довольно-таки громоздкой формулой:

$$\mathbf{v}_\perp = \mathbf{v}_1 \times \mathbf{v}_2,$$

$$\mathbf{m}_\perp = \mathbf{m}_1 \times \mathbf{l}_2 - \mathbf{m}_2 \times \mathbf{v}_1 +$$

$$+ \frac{(\mathbf{v}_1, \mathbf{m}_2) + (\mathbf{v}_2, \mathbf{m}_1)}{\mathbf{v}_1 \times \mathbf{v}_2} (\mathbf{v}_1, \mathbf{v}_2) \mathbf{v}_1 \times \mathbf{v}_2.$$

Компьютерная реализация имеет вид:

```
Line3D reciprocal_perp_line(Line3D line01,
↪  Line3D line02) {
  triple V = cross(line01.v, line02.v);
  triple m = cross(line01.m, line02.v) -
  ↪  cross(line02.m, line01.v) +
  ↪  reciprocal_perp_length(line01,
  ↪  line02) * dot(line01.v, line02.v) *
  ↪  cross(line01.v, line02.v) /
  ↪  length(cross(line01.v, line02.v));
  return Line3D(V=V, m=m);
}
```

4. ПРОЕКТИВНЫЕ ПЛОСКОСТИ В ВИДЕ СТРУКТУРЫ PLANE3D

4.1. Структура и конструкторы

Теперь рассмотрим структуру, реализующую задание плоскости в однородном виде. Для полного описания плоскости достаточно вектора $\vec{\boldsymbol{\pi}} = [\mathbf{n} \,|\, d]$, где $\mathbf{n}$ — единичный вектор нормали к плоскости, а d — ориентированное расстояние от начала координат до плоскости.

В модуле three языка Asymptote существует функция plane со следующей сигнатурой:

```
path3 plane(triple u, triple v,
↪  triple0=0);
```

Здесь u и v — направляющие векторы плоскости, а 0 — точка их приложения. Данная функция возвращает замкнутый прямоугольный контур 0 -- 0+u -- 0+u+v -- 0+v -- cycle, который затем можно отобразить в виде плоскости с помощью функции surface. Поэтому в структуре Plane3D полезно хранить два направляющих вектора $\mathbf{v}$ и $\mathbf{u}$, а также произвольную точку P. Полностью поля структуры будут выглядеть следующим образом:

```
struct Plane3D {
  // Вектор нормали (ненормированный):
  triple N;
  // Единичный вектор нормали:
  triple n;
  // Направляющие векторы (касательные
  ↪  векторы):
  triple U, V;
  // Единичные направляющие векторы:
  triple u, v;
  // Ориентированное расстояние от
  ↪  начала координат до прямой:
  real d;
  // Произвольная точка плоскости:
  triple P;
}
```

Зададим несколько сравнительно тривиальных операторов инициализации. Первый из них создает плоскость, содержащую прямую $\vec{\mathbf{l}} = \{\mathbf{v} \mid \mathbf{n}\}$ и направляющий вектор $\mathbf{v}$:

```
void operator init (Line3D line, triple
↪  U) {
  this.U = U;
  this.V = line.V;

  this.u = unit(U);
  this.v = line.v;

  this.N = cross(this.v, this.u);
  this.n = unit(this.N);
  this.d = -dot(this.u, line.m);

  this.P = -this.d * this.n;
}
```

В качестве точки плоскости берется ближайшая к началу координат. Вычисление проводится по формуле

$$[\mathbf{v} \times \mathbf{u} \mid -(\mathbf{u}, \mathbf{m})].$$

Второй конструктор инициализирует плоскость, проходящую через точку и прямую. Точка может быть задана однородными координатами, тогда работает формула

$$[\mathbf{v} \times \mathbf{p} + w\mathbf{m} \mid -(\mathbf{p}, \mathbf{m})].$$

Точка может быть задана декартовыми координатами, тогда вычисления проводятся по формуле

$$[\mathbf{v} \times \mathbf{p} + \mathbf{m} \mid -(\mathbf{p}, \mathbf{m})].$$

Так как случай однородных координат более общий, то мы ограничимся только им в реализации конструктора:

```
void operator init (Line3D line, Vec4 P)
↪  {
  this.U = line.P - P.xyz;
  this.V = line.V;
```

```
  this.u = unit(U);
  this.v = line.v;

  this.N = cross(line.v, P.xyz) + P.w *
  ↪  line.m;
  this.n = unit(this.N);
  this.d = -dot(P.xyz, line.m);

  this.P = P.xyz;
}
```

Третий конструктор инициализирует плоскость, проходящую через прямую и начало координат:

```
void operator init (Line3D line) {
  this.U = line.P;
  this.V = line.V;

  this.u = unit(U);
  this.v = line.v;

  this.N = line.m;
  this.n = unit(this.N);
  this.d = 0;

  this.P = line.P;
}
```

Вычисления проводятся по следующей формуле:

$$[\mathbf{m} \mid 0].$$

Четвертый конструктор создает плоскость, проходящую через заданную точку и два направляющих вектора. Это эквивалентно записи параметрического уравнения плоскости. Вычисления проводятся по формуле (8). В Asymptote нет смешанного произведения, но оно естественно заменяется на комбинацию скалярного и векторного произведений:

```
void operator init (triple P, triple U,
↪  triple V) {
  this.U = U;
  this.V = V;

  this.u = unit(U);
  this.v = unit(V);

  this.N = cross(U, V);
  this.n = unit(this.N);
  this.d = dot(V, cross(U, P)) /
  ↪  length(this.N);

  this.P = P;
}
```

Наконец пятый конструктор — более сложный по сравнению с двумя предыдущими. Он задает плоскость по коэффициентам общего уравнения. В этом случае довольно просто вычислить единичный вектор нормали и ориентированное расстояние от центра координат, но не так просто получить пару направляющих векторов:

```
void operator init (real A, real B, real
↪  C, real D) {
  this.N = (A, B, C);
  this.n = unit(this.N);
  triple w;
  if ( A <= B && A <= C) {
    w = this.n + (1, 0, 0);
  } else if (B <= A && B <= C) {
    w = this.n + (0, 1, 0);
  } else {
    w = this.n + (0, 0, 1);
  }
  this.u = unit(cross(w, this.n));
  this.v = cross(this.n, this.u);
  this.U = u;
  this.V = v;
  this.d = D / length(this.N);
  this.P = -this.d * this.n;
}
```

По известному единичному нормальному вектору плоскости $\mathbf{n}$ можно вычислить направляющие векторы плоскости $\mathbf{u}$ и $\mathbf{v}$. Для этого можно использовать простую процедуру, указанную в [20, с. 29]. Возьмем произвольный вектор $\mathbf{w}$, не коллинеарный $\mathbf{n}$. Чтобы найти такой вектор, можно взять наименьшую компоненту вектора $\mathbf{n}$ и увеличить ее на 1, например $\mathbf{w} = (n_x, n_y, n_z + 1)$, если $n_z < n_x$ и $n_z < n_y$. После этого можно вычислить вектор u:

$$\mathbf{u} = \frac{\mathbf{w} \times \mathbf{n}}{\|\mathbf{w} \times \mathbf{n}\|}.$$

Из свойств векторного произведения следует, что $\mathbf{u} \perp \mathbf{n}$. При вычислениях с помощью компьютера полезно также удостовериться, что значение $\|\mathbf{w} \times \mathbf{n}\|$ не слишком велико или мало, чтобы избежать лишних погрешностей, связанных с машинной точностью.

На последнем шаге находим $\mathbf{v}$ как $\mathbf{v} = \mathbf{n} \times \mathbf{u}$. Так как $\|\mathbf{n}\| = \|\mathbf{u}\| = 1$, то и $\mathbf{v}$ — единичный вектор. Таким образом мы получили ортонормированную тройку векторов $\langle \mathbf{n}, \mathbf{u}, \mathbf{v} \rangle$.

Точку плоскости можно найти как проекцию начала координат на плоскость $\mathbf{OO}_\perp = -d\mathbf{n}$ при условии, что $\mathbf{n}$ — единичный вектор:

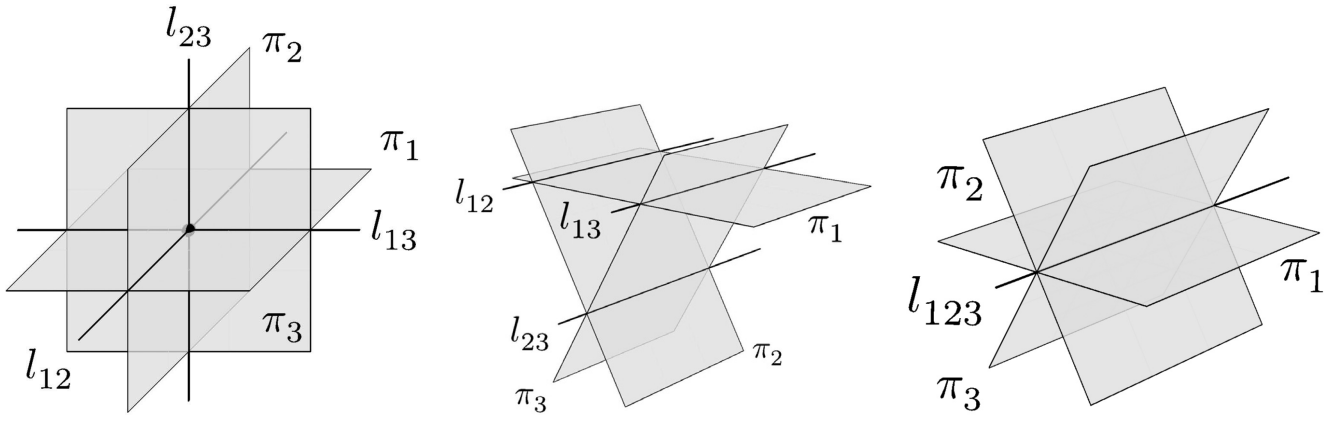


Рис. 1. Варианты непараллельных плоскостей.

$$(-d\mathbf{n} \mid \|\mathbf{n}\|^2).$$

В области имен структуры зададим еще две функции. Первая из них фактически является параметрическим уравнением плоскости:

```
triple get_point(real s, real t) {
  return this.P + this.u * s + this.v *
  ↪  t;
}
```

в виде прямоугольника:

```
guide3 get_contour() {
  triple P = this.P - 0.5(this.u +
  ↪  this.v);
  return  P -- P + this.u -- P + this.u
  ↪  + this.v -- P +this.v -- cycle;
}
```

При этом точка приложения направляющих векторов смещена так, чтобы визуально она лежала в центре прямоугольника, составляющего контур плоскости.

4.2. Взаимное расположение плоскостей

Имея в наличии структуру Plane3D, можно перейти к решению задачи определения взаимного положения двух и трех плоскостей.

В проективном пространстве две плоскости пересекаются всегда, так как параллельные плоскости считаются пересекающимися в бесконечности и прямая, по которой они пересекаются, является несобственной. Прямая пересечения вычисляется по формуле:

$$\{\mathbf{v} \mid \mathbf{m}\} = \{\mathbf{n}_1 \times \mathbf{n}_2 \mid d_1\mathbf{n}_2 - d_2\mathbf{n}_1\}. \qquad (6)$$

Возможны следующие варианты:

- если $\mathbf{v} \neq \mathbf{0}$, то прямая собственная;
- если $\mathbf{v} = \mathbf{0}$, то прямая несобственная;
  - если $d_1 = d_2$, то плоскости совпадают;
  - если $d_1 \neq d_2$, то плоскости параллельны.

Отметим также, что если две плоскости совпадают, то необязательно $\mathbf{n}_1 = \mathbf{n}_2$, так как возможен вариант, когда $\mathbf{n}_1 = -\mathbf{n}_2$. В этом случае лицевая и внутренняя стороны плоскостей отличаются.

В случае трех плоскостей вариантов может быть больше. Все они изображены на рис. 1 и 2.

Чтобы выписать формулы для вычислений, введем следующие обозначения. Плоскости π_1, π_2 и π_3 запишем в координатах Плюккера в следующем виде: $\vec{\boldsymbol{\pi}}_1 = [\mathbf{n}_1 | d_1]$, $\vec{\boldsymbol{\pi}}_2 = [\mathbf{n}_2 | d_2]$ и $\vec{\boldsymbol{\pi}}_3 = [\mathbf{n}_3 | d_3]$. Каждая пара этих трех плоскостей пересекается по некоторой прямой l_{ij}, где $i, j = 1, 2, 3$ и $i < j$.

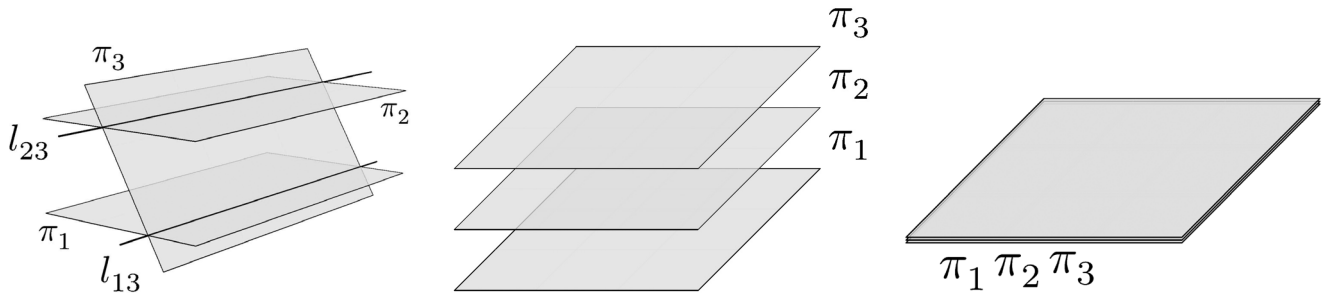
Тогда по формуле (14) запишем

$$\vec{\mathbf{l}}_{ij} = \{\mathbf{n}_i \times \mathbf{n}_j \mid d_i\mathbf{n}_j - d_j\mathbf{n}_i\}.$$

То есть получается следующее соотношение: $v_{ij} = \mathbf{n}_i \times \mathbf{n}_j$ и $\mathbf{m}_{ij} = d_1\mathbf{n}_2 - d_2\mathbf{n}_1$.

Все возможные варианты расположения трех плоскостей можно определить по следующей схеме:

- точка пересечения является собственной, т.е. $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3) \neq 0$;
- точка пересечения является несобственной, т.е. $(\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3) = 0$:
  - все прямые являются собственными, параллельными и различными (не совпадают), т.е. $\mathbf{v}_{12} \times \mathbf{v}_{13} = \mathbf{v}_{13} \times \mathbf{v}_{23} = \mathbf{v}_{23} \times \mathbf{v}_{12} = \mathbf{0}$ и $\mathbf{m}_{12} \neq \mathbf{m}_{23} \neq \mathbf{m}_{13}$;
  - все прямые являются собственными, параллельными и совпадают, т.е. $\mathbf{v}_{12} \times \mathbf{v}_{13} = \mathbf{v}_{13} \times \mathbf{v}_{23} = \mathbf{v}_{23} \times \mathbf{v}_{12} = \mathbf{0}$ и $\mathbf{m}_{12} = \mathbf{m}_{23} = \mathbf{m}_{13}$;



(а) (б) (в)

Рис. 2. Варианты параллельного расположения плоскостей. Вариант (в) — предельный случай варианта (б).

- две прямые являются собственными, параллельными и различными (не совпадают), т.е. $\mathbf{v}_{13} \times \mathbf{v}_{23} = \mathbf{0}$ и $\mathbf{m}_{13} \neq \mathbf{m}_{23}$, а одна прямая является несобственной, т.е. $\mathbf{v}_{12} = \mathbf{0}$;
- все прямые являются несобственными, т.е. $\mathbf{v}_{12} = \mathbf{v}_{23} = \mathbf{v}_{13} = \mathbf{0}$, а плоскости не совпадают, т.е. $d_1 \neq d_2 \neq d_3$;
- все прямые являются несобственными, т.е. $\mathbf{v}_{12} = \mathbf{v}_{23} = \mathbf{v}_{13} = 0$, а плоскости совпадают, т.е. $d_1 = d_2 = d_3$.

Однородные координаты точки пересечения трех плоскостей вычисляются по формуле:

$$p = \begin{pmatrix} d_1\mathbf{n}_3 \times \mathbf{n}_2 + d_2\mathbf{n}_1 \times \mathbf{n}_3 + \\ + d_3\mathbf{n}_2 \times \mathbf{n}_1 \mid (\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3) \end{pmatrix} = \\ = (\Delta_x, \Delta_y, \Delta_z \mid \Delta). \quad (15)$$

Обозначения Δ_x, Δ_y, Δ_z представляют собой определители решения системы уравнений $(\vec{\mathbf{n}}, \vec{\mathbf{p}}) = 0$ методом Краммера:

$$\begin{cases} n_{1x}x + n_{1y}y + n_{1z}z = -d_1, \\ n_{2x}x + n_{2y}y + n_{2z}z = -d_2, \\ n_{3x}x + n_{3y}y + n_{3z}z = -d_3. \end{cases}$$

$$\Delta = \begin{vmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ n_{3x} & n_{3y} & n_{3z} \end{vmatrix}, \quad \Delta_x = \begin{vmatrix} -d_1 & n_{1y} & n_{1z} \\ -d_2 & n_{2y} & n_{2z} \\ -d_3 & n_{3y} & n_{3z} \end{vmatrix},$$

$$\Delta_y = \begin{vmatrix} n_{1x} & -d_1 & n_{1z} \\ n_{2x} & -d_2 & n_{2z} \\ n_{3x} & -d_3 & n_{3z} \end{vmatrix}, \quad \Delta_z = \begin{vmatrix} n_{1x} & n_{1y} & -d_1 \\ n_{2x} & n_{2y} & -d_2 \\ n_{3x} & n_{3y} & -d_3 \end{vmatrix}.$$

Обозначение $(\Delta_x, \Delta_y, \Delta_z \mid \Delta)$ задает точку в однородных координатах:

$$\vec{\mathbf{p}} = \left(\frac{\Delta_x}{\Delta}, \frac{\Delta_y}{\Delta}, \frac{\Delta_z}{\Delta} \,\middle|\, 1 \right) =: (\Delta_x, \Delta_y, \Delta_z \mid \Delta).$$

Рассмотрим теперь реализацию указанных выше формул в виде кода. Можно встроить непосредственно в функции проверки того, являются ли получаемые точки и прямые несобственными. Однако если оперировать только проективными объектами, то такая проверка излишняя и ее можно проводить непосредственно в момент визуализации, извлекая всю информацию из однородного представления точек, прямых и плоскостей.

Вычисление общей прямой, по которой пересекаются две плоскости, можно реализовать следующим образом:

```
Line3D intersection(Plane3D plane01,
 ↪  Plane3D plane02) {
  Line3D line;
  line.m = plane01.d * plane02.n -
   ↪  plane02.d * plane01.n;
  line.V = cross(plane01.n, plane02.n);
  // Если плоскости пересекаются по
   ↪  несобственной прямой, то v = 0
  if (abs(line.V) < 1e-8) {
    line.v = (0, 0, 0);
  } else {
    line.v = unit(line.V);
    // проведем нормировку, разделив на
     ↪  длину направляющего вектора
    line.m = line.m / length(line.V);
    line.P = cross(line.v, line.m) /
     ↪  dot(line.v, line.v);
  }
  return line;
}
```

Необходимые проверки были встроены непосредственно в код. Функция abs вычисляет норму вектора (можно заменить на функцию length). Еще раз следует заметить, что в случае использования чисел с плавающей запятой необходимо учитывать

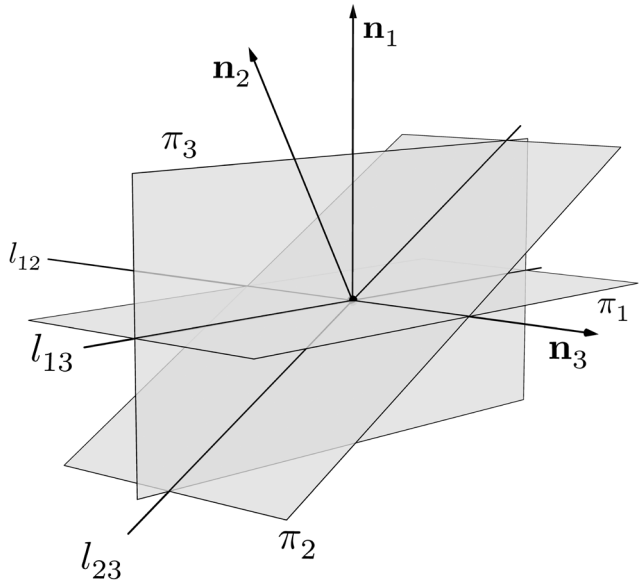


Рис. 3. Пересечение трех плоскостей.

машинную погрешность и использовать не строгое равенство, а проверять, что модуль числа не превосходит некоторое малое число.

Код для вычисления точки пересечения трех плоскостей будет выглядеть следующим образом:

```
Vec4 intersection(Plane3D plane01,
↪  Plane3D plane02, Plane3D plane03) {
  triple res;

  real n1n2n3 = dot(plane01.n,
  ↪  cross(plane02.n, plane03.n));

  res  = plane01.d * cross(plane03.n,
  ↪  plane02.n);
  res += plane02.d * cross(plane01.n,
  ↪  plane03.n);
  res += plane03.d * cross(plane02.n,
  ↪  plane01.n);
  return Vec4(res, n1n2n3);
}
```

В этом примере мы не делаем никаких проверок, так как никакая комбинация аргументов не приведет к исключительному случаю. Функция возвращает точку в однородных координатах. Проверку на собственность/несобственность можно сделать уже вне функции. В случае несобственной точки для дальнейшего анализа необходимо использовать функцию для определения прямых пересечения плоскостей.

Еще одна функция реализует формулу для вычисления однородных координат точки пересечения плоскости $[\mathbf{n} \mid d]$ и прямой $\{\mathbf{v} \mid \mathbf{m}\}$:

$$\left(-\frac{(\mathbf{m}\times\mathbf{n}+d\mathbf{v})}{(n,v)}\middle|1\right) = \left(-(\mathbf{m}\times\mathbf{n}+d\mathbf{v}) \mid (\mathbf{n},\mathbf{v})\right) = \\ = \left((\mathbf{m}\times\mathbf{n}+d\mathbf{v}) \mid -(\mathbf{n},\mathbf{v})\right). \quad (16)$$

Здесь также можно поступить двояко: вернуть тип triple и встроить все проверки внутрь функции, или же вернуть тип Vec4 и возложить ответственность за проверки на пользователя функции. Здесь мы приводим первый вариант функции:

```
triple intersection(Plane3D plane, Line3D
↪  line) {

  assert( dot(line.v, plane.n) > 1e-8,
  ↪  "Плоскость и прямая не
  ↪  пересекаются!");

  return -(cross(line.m, plane.n) +
  ↪  plane.d * line.v) / dot(line.v,
  ↪  plane.n);
}
```

5. ПРИМЕР ВИЗУАЛИЗАЦИИ

В данном разделе рассмотрим пример использования созданных структур и встроенных в Asymptote средств для визуализации комплексного изображения пересечения плоскостей и прямых. Приведем полный код программы и прокомментируем все его части.

Программа рисует три плоскости, которые пересекаются в одной точке. Изображаются прямые пересечения каждой пары плоскостей. Точка пересечения плоскостей одновременно является точкой пересечения прямых, по которым пересекаются данные плоскости. Результат показан на рис. 3.

Третья плоскость специально выбрана так, чтобы ее нормальный вектор совпадал с направляющим вектором прямой пересечения первой и второй плоскостей, т.е. $\mathbf{n}_3 = \mathbf{n}_1 \times \mathbf{n}_2 = \mathbf{v}_{12}$. Это изображение может служить иллюстрацией к доказательству формулы нахождения прямой, по которой пересекаются две плоскости.

Файл с исходным кодом Asymptote имеет расширение **.asy**. Для его запуска будем использовать команду **asy**. Вначале следует импортировать все используемые модули как встроенные, так и реализованные нами.

```
import settings;

settings.outformat = "pdf";
settings.render = 15;
```

```
include "Vec4.asy";
include "Line3D.asy";
include "Plane3D.asy";

import three;
import graph3;


unitsize(3cm);
```

Модуль settings позволяет настроить некоторые параметры работы Asymptote. В частности мы указываем, что следует создавать изображение в формате pdf, а также выставляем качество созданного трехмерного изображения. Значения больше нуля приведут к тому, что изображение получится растровым. В случае трехмерных чертежей мы вынуждены использовать растровый формат, так как иначе некорректно рассчитываются наложения поверхностей друг на друга. Качество растра следует подбирать опытным путем.

Далее импортируем три созданные нами структуры модуль three для поддержки трехмерных векторов и модуль graph3 для визуализации трехмерных поверхностей и кривых. Также выставляем единицу измерения по умолчанию в 3 сантиметра — такое значение было подобрано опытным путем исходя из критерия читаемости изображения при его публикации в статье или презентации.

В модуле three определена функция plane, которая принимает в качестве аргументов два направляющих вектора плоскости и точку их приложения, а в качестве результата возвращает контур плоскости в виде прямоугольника. Точка приложения векторов будет служить правым верхним углом данного прямоугольника. Далее этот контур можно использовать для отображения плоскости с помощью функции surface.

В первую очередь задаем направляющие векторы:

```
triple u_01 = X;
triple v_01 = Y;

triple u_02 = rotate(30, Y) * u_01;
triple v_02 = v_01;

triple u_03 = u_01;
triple v_03 = rotate(-90, X) * v_01;
```

Для первой плоскости в качестве направляющих выберем орты осей *Ox* и *Oy*, используя заданные в модуле graph3 векторы X, Y.

Точки приложения для каждой плоскости следует задавать отдельно, иначе все плоскости будут визуально исходить из одной точки, что сделает рисунок ненаглядным:

```
// Произвольная точка, к которой будем
↪  привязывать направляющие векторы
triple P_0 = (0, -0.5, 1);
// Точка плоскости, от которой будут
↪  откладываться направляющие векторы при
↪  рисовании

triple P_01 = P_0 - u_01;
triple P_02 = P_0 - u_02;
triple P_03 = P_0 - X + 0.5Y + 0.5Z;
```

Далее можно создать контуры и поверхности плоскостей:

```
// Создаем контуры плоскостей
path3 pl_01 = plane(u=2u_01, v=v_01,
↪  O=P_01);
path3 pl_02 = plane(u=2u_02, v=v_02,
↪  O=P_02);
path3 pl_03 = plane(u=2u_03, v=v_03,
↪  O=P_03);

surface pls_01 = surface(pl_01);
surface pls_02 = surface(pl_02);
surface pls_03 = surface(pl_03);
```

Чтобы созданные объекты были отрисованы, необходимо воспользоваться функцией draw. Покажем как отрисовать первую плоскость:

```
draw(
  s=pls_01,
  nu=1, nv=1,
  surfacepen=lightgrey+opacity(.8),
  meshpen=black+thin(),
  light=nolight
);
```

Разберем передаваемые в данную функцию параметры:

- в параметр s передается объект surface;
- параметры nu и nv задают количество точек сетки сплайновой поверхности;
- параметр surfacepen позволяет настроить перо для рисования поверхности;
- параметр meshpen отдельно настраивает перо для сетки, в нашем случае от сетки остается только контур плоскости;
- с помощью параметра light настраивается свет.

Мы хотим сделать подписи к элементам рисунка, в частности подписать обозначения для плоскостей. Для этого отображаем контуры плоскости отдельно:

```
draw(pl_01, p=0.25bp+black,
 ↪  L=Label("$\pi_1$",
 ↪  position=Relative(0.80)));
draw(pl_02, p=0.25bp+black,
 ↪  L=Label("$\pi_2$"));
draw(pl_03, p=0.25bp+black,
 ↪  L=Label("$\pi_3$",
 ↪  position=Relative(0.3)));
```

Здесь параметр p задает перо, параметр L — подпись к изображаемому объекту. Для создания подписей можно использовать нотацию LaTeX, что делает Asymptote удобным для создания иллюстраций к научным текстам.

Далее перейдем к использованию созданных нами структур. В первую очередь зададим все три плоскости:

```
Plane3D plane01 = Plane3D(P_01, u_01,
 ↪  v_01);
Plane3D plane02 = Plane3D(P_02, u_02,
 ↪  v_02);
Plane3D plane03 = Plane3D(P_03, u_03,
 ↪  v_03);
```

Затем найдем прямые пересечения данных плоскостей:

```
Line3D line12 = intersection(plane01,
 ↪  plane02);
Line3D line13 = intersection(plane01,
 ↪  plane03);
Line3D line23 = intersection(plane02,
 ↪  plane03);
```

Прямые отображаются в виде отрезков, поэтому следует знать две точки каждой прямой. Подбираются точки эмпирически, а чтобы получить координаты точек используем метод get_point. Например, для первой прямой:

```
triple l12_sp = line12.get_point(-1.6);
triple l12_ep = line12.get_point(+0.9);
```

Далее отображаем прямые. Так, для первой прямой:

```
draw(l12_sp--l12_ep, p=black,
 ↪  L=Label("$l_{12}$",
 ↪  position=BeginPoint));
```

Теперь находим точку пересечения и отображаем ее:

```
triple intersection_point =
 ↪  intersection(plane01, plane02,
 ↪  plane03);

dot(intersection_point);
```

Наконец, откладываем нормальные векторы всех трех плоскостей от точки их пересечения:

```
draw(
  intersection_point -- intersection_point
   ↪  + plane01.n,
  arrow=Arrow3(size=4),
  p=black,
  L=Label(s="$\vb{n}_1$", align=E,
   ↪  position=Relative(0.9))
);
```

6. ЗАКЛЮЧЕНИЕ

Мы рассмотрели реализацию аналитической проективной геометрии на языке Asymptote. Были реализованы однородные координаты, координаты Плюккера прямой и однородные координаты плоскости. Подробно описаны три созданные нами структуры, инициализирующие операторы и функции. Фундаментальным преимуществом по сравнению с классической аналитической геометрией является существенное упрощение вычислений, так как большинство функций просто повторяют проективные формулы и вычислительная часть зачастую умещается в несколько строк.

Следует особо отметить, что так как все функции Asymptote предназначены для работы с объектами трехмерного декартова пространства, то для окончательной визуализации полученных объектов мы все же вынуждены делать проверки на равенство нулю тех или иных величин, а также учитывать невозможность однозначно визуализировать несобственные точки, прямые и плоскости. Однако это уже ограничения технического характера.

Дальнейшая работа может иметь два направления. В математическом плане все используемые нами формулы могут быть записаны в терминах геометрической алгебры. С вычислительной точки зрения это не дает никаких преимуществ, однако с теоретической точки зрения дает более общий и гибкий взгляд на геометрические объекты и их взаимное расположение в пространстве.

Второе направление — техническое. Оно заключается в том, чтобы реализовать набор функций для

визуализации точек, прямых и плоскостей, заданных непосредственно в проективном виде.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Публикация выполнена в рамках проекта № 021934-0-000 Системы грантовой поддержки научных проектов РУДН (Геворкян М.Н., Королькова А.В., Севастьянов Л.А.) и при поддержке Программы стратегического академического лидерства РУДН (Кулябов Д.С.).

СПИСОК ЛИТЕРАТУРЫ

1. *Korolkova A.V., Gevorkyan M.N., Kulyabov D.S.* Implementation of hyperboliccomplex numbers in Julia language, Discrete Contin. Models Appl. Comput. Sci., 2022, vol. 30, no. 4, pp. 318–329.
2. *Kulyabov D.S., Korolkova A.V., Sevastianov L.A.* Complex numbers for relativistic operations, 2021.
3. *Kulyabov D.S., Korolkova A.V., Gevorkyan M.N.* Hyperbolic numbers as Einstein numbers, J Phys.: Conf. Ser., 2020, vol. 1557, p. 012027.
4. *Gevorkyan M.N., Korolkova A.V., Kulyabov D.S.* Approaches to the implementation of generalized complex numbers in the Julia language, Workshop on Information Technology and Scientific Computing in the framework of the X Int. Conf. Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems (ITTMM), Kulyabov, D.S., Samouylov, K.E., and Sevastianov, L.A., Eds., 2020, vol. 2639, pp. 141–157.
5. *Геворкян М.Н., Королькова А.В., Кулябов Д.С.* Реализация геометрической алгебры в системах символьных вычислений // Программирование. 2023. № 1. С. 48–55.
6. *Королькова А.В., Геворкян М.Н., Кулябов Д.С., Севастьянов Л.А.* Средства компьютерной алгебры для геометризации уравнений Максвелла // Программирование. 2023. Т. 49, № 4. С. 33–38.
7. *Велиева Т.Р., Геворкян М.Н., Демидова А.В. и др.* Аппарат геометрической алгебры и кватернионов в системах символьных вычислений для описания вращений в евклидовом пространстве // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 1. С. 31–42.
8. *Bowman J.C.* Hammerlindl A. Asymptote: A vector graphics language, 2008, vol. 29, no. 2, pp. 288–294.
9. *Bowman J.C.* Asymptote: Interactive TEX-aware 3D vector graphics, 2010, vol. 31, no. 2, pp. 203–205.
10. *Shardt O., Bowman J.C.* Surface parameterization of nonsimply connected planar Bzier regions, Comput.-Aided Des., 2012, vol. 44, no. 5, pp. 484.e1–484.e10.
11. *Bowman, J.C.* Asymptote: The vector graphics language, 2023. https://asymptote.sourceforge.io.
12. *Gevorkyan M.N., Korolkova A.V., Kulyabov D.S.* Asymptote-based scientific animation, Discrete Contin. Models Appl. Comput. Sci., 2023, vol. 31, no. 2, pp. 139–149.
13. *Страуструп Б.* Программирование. Принципы и практика с использованием C++. 2 изд. Вильямс, 2018. 1328 с.
14. *Hobby J., Knuth D.* MetaPost on the Web. https://www.tug.org/metapost.html.
15. *Staats C.* An Asymptote tutorial, 2022. https://asymptote.sourceforge.io/asymptote_tutorial.pdf.
16. *Крячков Ю.Г.* Asymptote для начинающих. http://mif.vspu.ru/books/ASYfb.pdf.
17. *Волченко Ю.М.* Научная графика на языке Asymptote. http://www.math.volchenko.com/AsyMan.pdf.
18. *Ивальди Ф.* Евклидова геометрия на языке векторной графики Asymptote.2015. http://mif.vspu.ru/books/geometry_new_ru.pdf.
19. *Lengyel E.* Foundations of game engine development, Terathon Software LLC, vol. 1. http://foundationsofgameenginedev.com.
20. *Marschner S., Shirley P.* Fundamentals of Computer Graphics, CRC Press, 5 ed.

IMPLEMENTATION OF ANALYTIC PROJECTIVE GEOMETRY FOR COMPUTER GRAPHICS

© 2024 M. N. Gevorkyan[a], A. V. Korol'kova[a], D. S. Kulyabov[a, b], L. A. Sevast'yanov[a, b]

[a] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198 Russia*
[b] *Joint Institute for Nuclear Research, 6 ul. Zholio-Kyuri 6, Dubna, Moscow oblast, 141980 Russia*

In their research, the authors actively exploit different branches of geometry. For geometric constructions, computer algebra approaches and systems are used. Currently, we are interested in computer geometry, more specifically, the implementation of computer graphics. The use of the projective space and homogeneous coordinates has actually become a standard in modern computer graphics. In other words, the problem is reduced to the application of analytic projective geometry. The authors failed to find a computer algebra system that could implement projective geometry in its entirety. Therefore, it was decided to partially implement computer algebra for visualization of algebraic relations. For this purpose, the Asymptote system was employed.

Keywords: projective geometry, Asymptote system, Plücker coordinates, proper and improper points, lines and planes

REFERENCES

1. *Korolkova A.V., Gevorkyan M.N., Kulyabov D.S.* Implementation of hyperboliccomplex numbers in Julia language, Discrete Contin. Models Appl. Comput. Sci., 2022, vol. 30, no. 4, pp. 318–329.
2. *Kulyabov D.S., Korolkova A.V., Sevastianov L.A.* Complex numbers for relativistic operations, 2021.
3. *Kulyabov D.S., Korolkova A.V., Gevorkyan M.N.* Hyperbolic numbers as Einstein numbers, J Phys.: Conf. Ser., 2020, vol. 1557, p. 012027.
4. *Gevorkyan M.N., Korolkova A.V., Kulyabov D.S.* Approaches to the implementation of generalized complex numbers in the Julia language, Workshop on Information Technology and Scientific Computing in the framework of the X Int. Conf. Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems (ITTMM), Kulyabov, D.S., Samouylov, K.E., and Sevastianov, L.A., Eds., 2020, vol. 2639, pp. 141–157.
5. *Gevorkyan M.N., Korol'kova A.V., Kulyabov D.S.* Implementation of geometric algebra in symbolic computation systems, Programmirovanie, 2023, no. 1, pp. 48–55.
6. *Korol'kova A.V., Gevorkyan M.N., Kulyabov D.S., Sevast'yanov L.A.* Computer algebra tools for geometrization of Maxwell's equations, Program. Comput. Software, 2023, vol. 49, pp. 336–371.
7. *Velieva T.R., Gevorkyan M.N., Demidova A.V., Korol'kova A.V., Kulyabov D.S.* Geometric algebra and quaternion techniques in computer algebra systems for describing rotations in Eucledean space, Comput. Math. Math. Phys., 2023, vol. 63, pp. 29–39.
8. *Bowman J.C.* Hammerlindl A. Asymptote: A vector graphics language, 2008, vol. 29, no. 2, pp. 288–294.
9. *Bowman J.C.* Asymptote: Interactive TEX-aware 3D vector graphics, 2010, vol. 31, no. 2, pp. 203–205.
10. *Shardt O., Bowman J.C.* Surface parameterization of nonsimply connected planar Bzier regions, Comput.-Aided Des., 2012, vol. 44, no. 5, pp. 484.e1–484.e10.
11. *Bowman, J.C.* Asymptote: The vector graphics language, 2023. https://asymptote.sourceforge.io.
12. *Gevorkyan M.N., Korolkova A.V., Kulyabov D.S.* Asymptote-based scientific animation, Discrete Contin. Models Appl. Comput. Sci., 2023, vol. 31, no. 2, pp. 139–149.
13. *Stroustrup B.* Programming: Principles and Practice Using C++, Addison-Wesley Professional, 2014, 2nd ed.
14. *Hobby J., Knuth D.* MetaPost on the web. https://www.tug.org/metapost.html.
15. *Staats C.* An Asymptote tutorial, 2022. https://asymptote.sourceforge.io/asymptote_tutorial.pdf.
16. *Kryachkov Yu.G.* Asymptote for beginners. http://mif.vspu.ru/books/ASYfb.pdf.
17. *Volchenko Yu.M.* Scientific graphics in the Asymptote language. http://www.math.volchenko.com/AsyMan.pdf.
18. *Ivaldi, Ph.*, Euclidean Geometry with ASYMPTOTE, 2011.
19. *Lengyel E.* Foundations of game engine development, Terathon Software LLC, vol. 1. http://foundationsofgameenginedev.com.
20. *Marschner S., Shirley P.* Fundamentals of Computer Graphics, CRC Press, 5 ed.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 66–73*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.422.8

СИМВОЛЬНЫЕ ИССЛЕДОВАНИЯ УРАВНЕНИЙ МАКСВЕЛЛА В ФОРМАЛИЗМЕ ПРОСТРАНСТВЕННО-ВРЕМЕННОЙ АЛГЕБРЫ

© 2024 г. А. В. Королькова[a], *, М. Н. Геворкян[a], **, А. В. Фёдоров[a], *, К. А. Штепа[a], ****, Д. С. Кулябов[a, b] *******

[a]*Российский университет дружбы народов, 117198 Москва, ул. Миклухо-Маклая, д. 6, Россия*
[b]*Объединенный институт ядерных исследований, 141980 Дубна, Московская обл., ул. Жолио-Кюри, д. 6, Россия*

**E-mail: korolkova-av@rudn.ru*
***E-mail: gevorkyan-mn@rudn.ru*
****E-mail: 1042210107@rudn.ru*
*****E-mail: 1042210111@pfur.ru*
******E-mail: kulyabov-ds@rudn.ru*

Поступила в редакцию 01.08.2023
После доработки: 10.09.2023
Принята к публикации 01.10.2023

Для описания физических и технических систем авторы используют разные реализации алгебры Клиффорда: спиноры, кватернионы, геометрическую алгебру. Формализм геометрической алгебры является сравнительно новым подходом, ориентированным в первую очередь на инженеров и прикладных исследователей. В целом ряде работ авторы рассмотрели реализацию формализма геометрической алгебры для систем компьютерной алгебры. В данной статье авторы расширяют эллиптическую геометрическую алгебру на гиперболическую пространственно-временную алгебру. В качестве иллюстрации используются разные представления уравнений Максвелла. С помощью системы компьютерной алгебры выполнен переход от вакуумных уравнений Максвелла в представлении пространственно-временной алгебры к уравнениям Максвелла в векторном формализме. Кроме практического применения, авторы хотели бы обратить внимание на дидактическое значение данных исследований.

Ключевые слова: геометрическая алгебра, алгебра Клиффорда, система компьютерной алгебры SymPy, Galgebra

DOI: 10.31857/S0132347424020078 **EDN:** ROSDXW

1. ВВЕДЕНИЕ

Разные реализации геометрической алгебры рассматривались авторами в работах [1] (обзор различных реализаций), [2] (реализация на языке Julia), [3] (реализации на основе библиотеки SymPy). В работе [4] демонстрировалось применение геометроалгебраического подхода к геометризованному представлению уравнений Максвелла [5, 6]. Все исследования реализовывались на основе систем компьютерной алгебры. Таким образом, данная работа продолжает наш цикл исследований. В данной статье рассматривается мультивекторная запись уравнений Максвелла.

Геометрическая алгебра основывается на работах Г. Г. Грассмана [7], У. Р. Гамильтона [8], У. К. Клиффорда [9]. В геометрической алгебре рассматривается конкретная реализация алгебры Клиффорда, основанная на мультивекторах, которая включает в себя реализацию алгебры Грассмана в виде внешней алгебры p-векторов (контравариантных антисимметричных тензоров с операцией внешнего умножения). Алгебра кватернионов также является частным случаем алгебры мультивекторов.

В статье символьные вычисления производятся с помощью модуля Galgebra [10, 11], основанного на пакете символьной алгебры SymPy [12, 13] для языка Python.

1.1. Структура статьи

В разделе 1.2 даются основные обозначения и соглашения, применяемые в статье. В разделе 2 излагаются основы алгебраической геометрии. Данное направление изучает ассоциативную алгебру мультивекторов, являющуюся реализацией абстрактной алгебры Клиффорда. Развитый в данной области математический аппарат пока не получил широкой известности, поэтому авторам показалось необходимым кратко изложить его основы. В разделе 3 по шагам производится переход от стандартного векторного формализма к формализму геометрической алгебры

на примере системы вакуумных уравнений Максвелла. В мультивекторной записи уравнения Максвелла сводятся к одному уравнению. В разделе 4 мы выполним в некотором роде обратную задачу, но теперь нашей целью будет демонстрация использования программных средств компьютерной алгебры.

1.2. Обозначения и соглашения

1. Будем придерживаться следующих соглашений. Греческие индексы α, β будут относиться к четырёхмерному пространству и в компонентном виде будут иметь следующие значения: $\alpha = \overline{0,3}$. Латинские индексы из середины алфавита i, j, k будут относиться к трёхмерному пространству и в компонентном виде будут иметь следующие значения: $i = \overline{1,3}$.

2. Контравариантные векторы (или просто векторы) будем обозначать строчными латинскими буквами и выделять с помощью полужирного шрифта, например $\mathbf{v}$, $\mathbf{x}$.

3. Линейное пространство контравариантных векторов будем обозначать буквой L и полагать существование базиса $\langle \mathbf{e}_1, \ldots, \mathbf{e}_n \rangle$. Нумерацию векторов базиса будем вести нижними индексами от 1 до n. Буквой n всегда будем обозначать размерность пространства L, т.е. $\dim L = n$.

4. Под полем скаляров будем подразумевать поле действительных чисел $\mathbb{R}$, хотя многие формулы и утверждения остаются справедливыми для любого произвольного поля R. Скаляры будем обозначать строчными греческими буквами α, β и т.д.

5. Будем полагать, что пространство L наделено структурой скалярного произведения и базис $\langle \mathbf{e}_1, \ldots, \mathbf{e}_n \rangle$ является ортонормированным.

6. Скалярное произведение векторов $\mathbf{u}$ и $\mathbf{v}$ будем обозначать $(\mathbf{u},\mathbf{v})$, векторное произведение — $[\mathbf{u},\mathbf{v}]$ или $\mathbf{u} \times \mathbf{v}$, смешанное произведение трех векторов — $(\mathbf{u},\mathbf{v},\mathbf{w})$.

7. Для записи уравнений электродинамики в работе используется система СИ [14]. Это вызвано тем, что формализм геометрической алгебры и пространственно-временной алгебры ориентируется, в первую очередь, на инженеров.

2. ГЕОМЕТРИЧЕСКАЯ АЛГЕБРА ПРОСТРАНСТВА-ВРЕМЕНИ

2.1. Основные понятия геометрической алгебры

Геометрическая алгебра является реализацией абстрактной алгебры Клиффорда [9], где элементами являются *мультивекторы*, а операцией "умножения" — операция *геометрического умножения*.

Мультивектор является градуированным объектом, представляющим собой линейную комбинацию разложимых p-векторов (кососимметричных ковариантных тензоров). Сами p-векторы совместно с операцией внешнего произведения $\wedge$ представляют собой реализацию абстрактной алгебры Грассмана [7]. Основным отличительным свойством внешнего произведения векторов является свойство *кососимметричности*: $\mathbf{v} \wedge \mathbf{v} = 0$. Более подробно о внешней алгебре можно прочитать в фундаментальных работах [15–21].

Рассмотрим четырехмерное пространство Минковского (пространство-время) $E^4_{1,3}$ с базисом $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$, метрическим тензором $g = \mathrm{diag}(1,-1,-1,-1)$ (иначе говоря с сигнатурой $(+,-,-,-)$):

$$(\mathbf{e}_0, \mathbf{e}_0) = 1, \quad (\mathbf{e}_0, \mathbf{e}_i) = 0, \quad (\mathbf{e}_i, \mathbf{e}_j) = -\delta_{ij}.$$

Операцию *геометрического умножения* для двух векторов $\mathbf{u}, \mathbf{v} \in E^4_{1,3}$ можно определить конструктивно с помощью формулы:

$$\mathbf{uv} = (\mathbf{u},\mathbf{v}) + \mathbf{u} \wedge \mathbf{v},$$

где сама операция не обозначается никаким знаком. Результатом действия геометрического произведения является элемент градуированной алгебры Клиффорда, называемой *геометрической алгеброй*.

Работа с геометрической алгеброй упрощается, если введен ортогональный базис. В случае пространства Минковского из определения геометрического умножения выводятся два ключевых свойства для базисных векторов:

$$\mathbf{e}_\alpha \mathbf{e}_\beta = -\mathbf{e}_\beta \mathbf{e}_\alpha, \ \alpha \neq \beta \quad \text{и} \quad \mathbf{e}_i \mathbf{e}_i = -1, \ \mathbf{e}_0 \mathbf{e}_0 = +1.$$

Также $\mathbf{e}_\alpha \wedge \mathbf{e}_\beta = \mathbf{e}_\alpha \mathbf{e}_\beta$, $\mathbf{e}_\alpha \wedge \mathbf{e}_\beta \wedge \mathbf{e}_\gamma = \mathbf{e}_\alpha \mathbf{e}_\beta \mathbf{e}_\gamma$ и т.д. Часто используют обозначение $\mathbf{e}_\alpha \mathbf{e}_\beta = \mathbf{e}_{\alpha\beta}$ и т.д.

В пространстве-времени операция внешнего произведения порождает четыре внешние алгебры:

- алгебру 1-векторов с базисом $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$;
- алгебру 2-векторов (бивекторов) с базисом $\langle \mathbf{e}_0\mathbf{e}_1, \mathbf{e}_0\mathbf{e}_2, \mathbf{e}_0\mathbf{e}_3, \mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \mathbf{e}_2\mathbf{e}_3 \rangle$;
- алгебру 3-векторов с базисом $\langle \mathbf{e}_{012}, \mathbf{e}_{013}, \mathbf{e}_{023}, \mathbf{e}_{123} \rangle$;
- алгебру 4-векторов (квадривекторов) с одним базисным 4-вектором $\mathbf{e}_{0123}$, который обычно обозначают как $\mathbf{I}$.

Шесть бивекторных базисов распадаются на два класса — пространственно-временной базис $\{\mathbf{e}_0\mathbf{e}_1, \mathbf{e}_0\mathbf{e}_2, \mathbf{e}_0\mathbf{e}_3\}$ и чисто пространственный базис $\mathbf{e}_1\mathbf{e}_2, \mathbf{e}_1\mathbf{e}_3, \mathbf{e}_2\mathbf{e}_3$. Пространственно-временные базисы обладают свойством гиперболической мнимой единицы, т.е. $\mathbf{e}_{0i}\mathbf{e}_{0i} = 1$, а чисто пространственные — свойством эллиптической мнимой единицы, т.е. $\mathbf{e}_{ij}\mathbf{e}_{ij} = -1$.

Можно составить следующие таблицы геометрического умножения:

	$\mathbf{e}_{01}$	$\mathbf{e}_{02}$	$\mathbf{e}_{03}$
$\mathbf{e}_{01}$	1	$-\mathbf{I}\mathbf{e}_{03}$	$\mathbf{I}\mathbf{e}_{02}$
$\mathbf{e}_{02}$	$\mathbf{I}\mathbf{e}_{03}$	1	$-\mathbf{I}\mathbf{e}_{01}$
$\mathbf{e}_{01}$	$-\mathbf{I}\mathbf{e}_{02}$	$\mathbf{I}\mathbf{e}_{01}$	1

	$\mathbf{e}_{12}$	$\mathbf{e}_{13}$	$\mathbf{e}_{23}$
$\mathbf{e}_{12}$	-1	$\mathbf{e}_{23}$	$-\mathbf{e}_{13}$
$\mathbf{e}_{13}$	$-\mathbf{e}_{23}$	-1	$\mathbf{e}_{12}$
$\mathbf{e}_{23}$	$\mathbf{e}_{13}$	$-\mathbf{e}_{12}$	-1

Из первой таблицы видно, что пространственно-временные базисы изоморфны матрицам Паули (с точности до знаков), поэтому в [19, стр. 135] для обозначения $\mathbf{e}_{0i}$ используется буква сигма σ_i.

2.2. Дифференциальные операторы

Введем оператор *векторной производной* (vector derivative) [19, стр. 168], определяемый следующей формулой:

$$\nabla = \mathbf{e}^i \frac{\partial}{\partial x^i} = \mathbf{e}^1 \frac{\partial}{\partial x^{i1}} + \ldots + \mathbf{e}^n \frac{\partial}{\partial x^n},$$

где $\{\mathbf{e}^i\}$ — *взаимный базис* (reciprocal frame), векторы которого определяются следующим образом:

$$(\mathbf{e}^i, \mathbf{e}_j) = \delta^i_j.$$

Для произвольного вектора $\mathbf{x}$ выполняется равенство

$$(\mathbf{e}^i, \mathbf{x}) = (\mathbf{e}^i, x^j \mathbf{e}_j) = x^j (\mathbf{e}^i, \mathbf{e}_j) =$$
$$= x^j \delta^i_j = x^i \Rightarrow x^i = (\mathbf{e}^i, \mathbf{x}).$$

В случае ортонормированного евклидова пространства $\mathbf{e}^i = \mathbf{e}_i$. В случае трехмерного или двумерного ортонормированного евклидова пространства оператор ∇ совпадает с классическим дифференциальным оператором $\vec{\nabla}$. С помощью ∇ и геометрического умножения можно записать операции градиента, дивергенции и ротора классического векторного анализа, а также обобщить их на большие размерности и метрики, отличные от ортонормированной.

Умножив геометрически ∇ на скалярную функцию $f(\mathbf{x})$, где $\mathbf{x} = x^i \mathbf{e}_i = x^1 \mathbf{e}_1 + \ldots + x^n \mathbf{e}_n$, получим формулу для градиента:

$$\nabla f(\mathbf{x}) = \mathbf{e}^i \frac{\partial f}{\partial x^i} = \left(\frac{\partial f}{\partial x^1}, \ldots, \frac{\partial f}{\partial x^n} \right),$$

который представляет собой вектор, записанный в взаимном базисе $\langle \mathbf{e}^1, \ldots, \mathbf{e}^n \rangle$.

Умножим ∇ на векторное поле $\mathbf{u}(\mathbf{x})$ и получим сумму скалярной и бивекторной частей:

$$\nabla \mathbf{u} = (\nabla, \mathbf{u}) + \nabla \wedge \mathbf{u}.$$

Скалярная часть является обобщением оператора дивергенции. Используем равенство $x^i = (\mathbf{e}^i, \mathbf{x})$ и запишем:

$$\left(\nabla, \mathbf{u}(\mathbf{x})\right) = \left(\frac{\partial}{\partial x^i} \mathbf{e}^i, \mathbf{u} \right) =$$
$$= \frac{\partial}{\partial x^i} (\mathbf{e}^i, \mathbf{u}) = \frac{\partial u^i}{\partial x^i} = \frac{\partial u^1}{\partial x^1} + \ldots + \frac{\partial u^n}{\partial x^n}.$$

Бивекторная часть обобщает операцию ротора:

$$\nabla \wedge \mathbf{u} = \mathbf{e}^i \wedge \frac{\partial \mathbf{u}}{\partial x^i} = \frac{\partial u^j}{\partial x^i} \mathbf{e}^i \wedge \mathbf{e}_i.$$

3. МУЛЬТИВЕКТОРНАЯ ЗАПИСЬ УРАВНЕНИЙ МАКСВЕЛЛА

Получим мультивекторную запись вакуумных уравнений Максвелла.

Уравнения Максвелла в дифференциальной форме в системе СИ имеют следующий вид:

$$\begin{cases} \vec{\nabla} \times \mathbf{H} = \dfrac{\partial \mathbf{D}}{\partial t} + \mathbf{j}, \\ \vec{\nabla} \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \vec{\nabla} \cdot \mathbf{D} = \rho, \\ \vec{\nabla} \cdot \mathbf{B} = 0. \end{cases} \tag{1}$$

Здесь $\mathbf{B}$ — вектор индукции магнитного поля, $\mathbf{E}$ — вектор напряженности электрического поля, $\mathbf{H}$ — напряжённость магнитного поля, $\mathbf{D}$ — электрическая индукция, $\mathbf{j}$ — вектор плотности внешнего электрического тока, ρ — плотность электрического заряда.

3.1. Уравнения Максвелла в дифференциальной форме для изотропной среды

Диэлектрическую и магнитную проницаемости положим равными единице, т.е. $\varepsilon = \mu = 1$ (вакуум). Кроме того, будем считать, что

$$\begin{aligned} \mathbf{H} &= \mathbf{B}, \\ \mathbf{D} &= \mathbf{E}. \end{aligned} \tag{2}$$

Тогда вакуумные уравнения Максвелла в дифференциальной форме (1) приобретают вид:

$$\begin{cases} \vec{\nabla} \times \mathbf{B} = \dfrac{\partial \mathbf{E}}{\partial t} + \mathbf{j}, \\ \vec{\nabla} \times \mathbf{E} = -\dfrac{\partial \mathbf{B}}{\partial}, \\ \vec{\nabla} \cdot \mathbf{E} = \rho, \\ \vec{\nabla} \cdot \mathbf{B} = 0. \end{cases} \tag{3}$$

При этом для плотностей тока **j** и заряда ρ выполняется уравнение непрерывности:

$$\vec{\nabla} \cdot \mathbf{j} + \frac{\partial \rho}{\partial t} = 0.$$

Символом ∇ в данной системе обозначен оператор набла, с помощью которого записаны операторы ротора $\vec{\nabla} \times$ и дивергенции $\vec{\nabla} \cdot$. Далее данный символ будет выступать только в роли оператора векторной производной.

Отметим, что все векторы в данной записи уравнений Максвелла являются векторами трехмерного Евклидова пространства (а точнее векторными полями, зависящими явно от трех координат и неявно от времени *t*). Их компоненты обозначаются с нижними индексами *x*,*y*,*z*, например E_x, E_y, E_z.

3.2. Мультивектор Фарадея

Рассмотрим теперь необходимые элементы, с помощью которых можно записать уравнения Максвелла в мультивекторном виде. Для этого введем пространство Минковского с сигнатурой (+,−,−,−) и базисными векторами $\langle \mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \rangle$. Оператор векторной производной в этом базисе записывается следующим образом:

$$\nabla = \mathbf{e}_0 \frac{\partial}{\partial x^0} - \mathbf{e}_1 \frac{\partial}{\partial x^1} - \mathbf{e}_2 \frac{\partial}{\partial x^2} - \mathbf{e}_3 \frac{\partial}{\partial x^3},$$

так как взаимный базис $\{\mathbf{e}^\alpha\}$, $\alpha = 0, 1, 2, 3$ имеет вид $\langle \mathbf{e}_0, -\mathbf{e}_1, -\mathbf{e}_2, -\mathbf{e}_3 \rangle$, в чем можно убедиться, вычислив скалярные произведения:

$$(\mathbf{e}_0, \mathbf{e}_0) = 1,$$
$$(\mathbf{e}_1, -\mathbf{e}_1) = 1,$$
$$(\mathbf{e}_2, -\mathbf{e}_2) = 1,$$
$$(\mathbf{e}_3, -\mathbf{e}_3) = 1.$$

Единичный элемент объема $\mathbf{e}_0\mathbf{e}_1\mathbf{e}_2\mathbf{e}_3$ обозначим как **I**.

Напряженность электрического поля **E** представим в виде бивектора с тремя пространственно-временными компонентами:

$$\mathbf{E} = E^{10}\mathbf{e}_1\mathbf{e}_0 + E^{20}\mathbf{e}_2\mathbf{e}_0 + E^{30}\mathbf{e}_3\mathbf{e}_0.$$

Бивекторные компоненты E^{10}, E^{20}, E^{30} соответствуют координатам вектора E_x, E_y, E_z классической векторной записи. Хотя у бивектора в четырехмерном пространстве Минковского может быть шесть ненулевых компонент, бивектор **E** имеет только три указанные ненулевые компоненты.

Аналогично представим индукцию магнитного поля **B**:

$$\mathbf{B} = B^{10}\mathbf{e}_1\mathbf{e}_0 + B^{20}\mathbf{e}_2\mathbf{e}_0 + B^{30}\mathbf{e}_3\mathbf{e}_0 = B_x\mathbf{e}_1\mathbf{e}_0 + B_y\mathbf{e}_2\mathbf{e}_0 + B_z\mathbf{e}_3\mathbf{e}_0.$$

Объединим плотность электрического заряда ρ (скаляр) и плотность внешнего электрического тока **j** (вектор) в четырехмерный пространственно-временной вектор **J**. Вектор **J** будем называть пространственно-временной плотностью электрического тока [19, стр. 230] и запишем его в следующем виде:

$$\mathbf{J} = \rho\mathbf{e}_0 + j^1\mathbf{e}_1 + j^2\mathbf{e}_2 + j^3\mathbf{e}_3 = \rho\mathbf{e}_0 + j_x\mathbf{e}_1 + j_y\mathbf{e}_2 + j_z\mathbf{e}_3.$$

Если умножить **J** справа на $\mathbf{e}_0$, то получим мультивектор со скалярной и бивекторной частями:

$$\mathbf{J}\mathbf{e}_0 = \rho + j_x\mathbf{e}_1\mathbf{e}_0 + j_y\mathbf{e}_2\mathbf{e}_0 + j_z\mathbf{e}_3\mathbf{e}_0.$$

Следовательно, плотность электрического тока **j** можно представить в виде бивектора $j^{10}\mathbf{e}_1\mathbf{e}_0 + j^{20}\mathbf{e}_2\mathbf{e}_0 + j^{30}\mathbf{e}_3\mathbf{e}_0$ с пространственно-временными компонентами (по аналогии с представлением **E** и **B**).

Наконец составим бивектор Фарадея **F** как сумму бивекторов **E** и **IB**:

$$\mathbf{F} = \mathbf{E} + \mathbf{IB}.$$

При умножении на **I** пространственно-временные бивекторные базисы $\mathbf{e}_0\mathbf{e}_1$, $\mathbf{e}_0\mathbf{e}_2$, $\mathbf{e}_0\mathbf{e}_3$ превращаются в чисто пространственные:

$$\mathbf{I}\mathbf{e}_1\mathbf{e}_0 = -\mathbf{e}_2\mathbf{e}_3,\ \mathbf{I}\mathbf{e}_2\mathbf{e}_0 = +\mathbf{e}_1\mathbf{e}_3,\ \mathbf{I}\mathbf{e}_3\mathbf{e}_0 = -\mathbf{e}_1\mathbf{e}_2,$$

из-за чего бивектор Фарадея имеет шесть компонент — максимально возможное количество компонент в четырехмерном пространстве Минковского:

$$\mathbf{F} = E^{10}\mathbf{e}_1\mathbf{e}_0 + E^{20}\mathbf{e}_2\mathbf{e}_0 + E^{30}\mathbf{e}_3\mathbf{e}_0 - B^{10}\mathbf{e}_2\mathbf{e}_3 + B^{20}\mathbf{e}_1\mathbf{e}_3 - B^{30}\mathbf{e}_3\mathbf{e}_2.$$

Теперь уравнения Максвелла можно записать в очень компактной форме:

$$\nabla\mathbf{F} = \mathbf{J}.$$

4. ВЫВОД ДИФФЕРЕНЦИАЛЬНОЙ ФОРМЫ УРАВНЕНИЙ МАКСВЕЛЛА ИЗ МУЛЬТИВЕКТОРНОЙ С ПОМОЩЬЮ СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ

Используем библиотеку Galgebra для того, чтобы вычислить компоненты мультивектора $\nabla\mathbf{F} - \mathbf{J}$ в декартовых координатах.

Вначале импортируем все необходимые модули:

```
import sympy as sp
from galgebra.ga import Ga
from galgebra.printer import latex #
↪ функция нужна для распечатки исходного
↪ кода LaTeX формул
from IPython.display import Math,
↪ DisplayObject # Функции нужны для
↪ отображения обработки и отображения
↪ LaTeX формул

sp.init_printing(latex_printer=latex,
↪ use_latex='mathjax',
↪ use_unicode=True)
```

Все вычисления мы будем проводить в интерактивной оболочке Jupyter Notebook [22]. Настроим отображение формул с помощью вызова следующей функции:

```
m4d = Ga('e', g=[1,-1,-1,-1],
↪ coords=sp.symbols('t, x, y,
↪ z',real=True))
```

После всех настроек зададим пространство Минковского и структуру геометрической алгебры на нем:

```
(grad, rgrad) = m4d.grads()
```

Здесь мы указываем символ **e**, который будет использоваться для обозначения базисного вектора, а также символы **t**, **x**, **y**, **z**, используемые для обозначения индексов. Также в параметре **g** указывается метрика, которая может быть только диагональной, но не обязательно нормированной.

Далее в отдельные переменные записываем базисные векторы, пространственно-временную часть бивекторного базиса и квадривекторный базисный элемент **I**:

```
# Пространственно временная часть
↪ бивекторного базиса
σ1 = e10 = e1*e0
σ2 = e20 = e2*e0
σ3 = e30 = e3*e0

I = e0*e1*e2*e3

Ex = sp.Function('E_1')(t,x,y,z)
```

Греческие буквы можно вводить в блокноте Jupyter с помощью их обозначений в формате LaTeX. Для этого следует набрать, например, **\sigma** и нажать клавишу **Tab**.

Для использования оператора векторной производной следует вызвать метод **grads** объекта **m4d** класса **Ga**:

```
(grad, rgrad) = o3d.grads()
```

Этот метод возвращает правый и левый операторы векторной производной, так как геометрическое умножение не коммутативно и при умножении слева следует использовать **grad**, а справа **rgrad**. Их значения равны соответственно:

$$\mathbf{e}_t\frac{\partial}{\partial t}-\mathbf{e}_x\frac{\partial}{\partial x}-\mathbf{e}_y\frac{\partial}{\partial y}-\mathbf{e}_z\frac{\partial}{\partial z},$$

$$\frac{\partial}{\partial t}\mathbf{e}_t-\frac{\partial}{\partial x}\mathbf{e}_x-\frac{\partial}{\partial y}\mathbf{e}_y-\frac{\partial}{\partial z}\mathbf{e}_z.$$

Такое разграничение двух операторов имеет смысл только в рамках модуля **Galgebra** в силу ограничений синтаксиса языка Python.

Далее задаем компоненты электрического и магнитного полей, вектор тока и плотность заряда как функции от переменных t, x, y, z:

```
Ez = sp.Function('E_3')(t,x,y,z)

Bx = sp.Function('B_1')(t,x,y,z)
By = sp.Function('B_2')(t,x,y,z)
Bz = sp.Function('B_3')(t,x,y,z)

jx = sp.Function('j_1')(t,x,y,z)
jy = sp.Function('j_2')(t,x,y,z)
jz = sp.Function('j_3')(t,x,y,z)

ρ = sp.Function('ρ')(t,x,y,z)

# напряженность электрического поля
```

После этого можно записать **E**, **B**, **j** и ρ:

```
# индукция магнитного поля
B = Bx*σ1 + By*σ2 + Bz*σ3
# плотность внешнего электрического тока
j = jx*σ1 + jy*σ2 + jz*σ3
# Пространственно-временной вектор тока
J = ρ*e0 + jx*e1 + jy*e2 + jz*e3

# Бивектор Фарадея
```

Теперь можно составить бивектор Фарадея:

```
# Бивектор Фарадея
```

В результате получим следующий бивектор с полным набором как пространственных, так и пространственно-временных компонент:

$$\mathbf{F} = -\mathbf{E}_x\mathbf{e}_{tx} - \mathbf{E}_y\mathbf{e}_{ty} - \mathbf{E}_z\mathbf{e}_{tz} - \\ - \mathbf{B}_z\mathbf{e}_{xy} + \mathbf{B}_y\mathbf{e}_{xz} - \mathbf{B}_x\mathbf{e}_{yz}.$$

Запишем теперь уравнения Максвелла в виде мультивектора:

```
# - уравнение непрерывности из равенства
↪  нулю скалярной части получившегося
↪  мультивектора;
```

Данный мультивектор имеет ненулевую векторную и тривекторную части. Приравняв весь мультивектор к нулю, мы получим дифференциальную форму уравнений Максвелла, если распишем отдельные компоненты данного мультивектора.

Векторную часть можно вычленить, вызвав метод **grade**:

```
Eq.grade(1)
```

Эта часть обладает следующими компонентами:

$$-\rho + \frac{\partial E_x}{\partial x}x + \frac{\partial E_y}{\partial y} + \frac{\partial E_z}{\partial z},$$
$$-j_x - \frac{\partial B_y}{\partial z} + \frac{\partial B_z}{\partial y} - \frac{\partial E_x}{\partial t},$$
$$-j_y + \frac{\partial B_x}{\partial y} + \frac{\partial B_z}{\partial x} - \frac{\partial E_y}{\partial t},$$
$$-j_z - \frac{\partial B_x}{\partial y} + \frac{\partial B_y}{\partial x} - \frac{\partial E_z}{\partial t}.$$

Приравнивание полученных компонент векторной части к нулю даст нам первое и третье уравнения из системы (3). В свою очередь, тривекторная часть получается вызовом метода **Eq.grade(3)** и также имеет четыре компоненты:

$$-\frac{\partial B_z}{\partial t} + \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x},$$
$$\frac{\partial B_y}{\partial t} + \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x},$$
$$\frac{-\partial B_x}{\partial t} + \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y},$$
$$\frac{\partial B_x}{\partial x} + \frac{\partial B_y}{\partial y} + \frac{\partial B_z}{\partial z},$$

приравняв которые к нулю мы получим второе и четвертое уравнения из (3).

Взяв повторно векторную производную:

$$\nabla\nabla\mathbf{F} - \nabla\mathbf{J} = 0, \qquad (4)$$

получим мультивектор со скалярной и полной бивекторной частями (с шестью компонентами).

Равенство нулю скалярной части выражения (4) дает уравнение непрерывности. Равенство нулю пространственно-временных компонент в выражении (4) дает волновое уравнение для электрического поля. Равенство нулю чисто пространственных компонент выражения (4) дает волновое уравнение для магнитного поля.

5. ЗАКЛЮЧЕНИЕ

В статье была рассмотрена мультивекторная запись уравнений Максвелла. Работу следует рассматривать как продолжение цикла исследований по реализациям геометрической алгебры и пространственно-временной алгебры для систем компьютерной алгебры. Статья носит не только прикладной, но и учебно-методический характер, поскольку в ней кроме демонстрации инструментального применения систем компьютерной алгебры рассмотрен вопрос получения с их помощью новых (не столь сложных, но скорее непривычных) математических формализмов для решения конкретных задач.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Публикация выполнена в рамках проекта № 021934-0-000 Системы грантовой поддержки научных проектов РУДН (Королькова А.В., Геворкян М.Н.) и при поддержке Программы стратегического академического лидерства РУДН (Кулябов Д.С., Фёдоров А.В., Штепа К.А.).

СПИСОК ЛИТЕРАТУРЫ

1. *Геворкян М.Н., Королькова А.В., Кулябов Д.С.* Реализация геометрической алгебры в системах символьных вычислений // Программирование. 2023. № 1. С. 48–55.
2. *Геворкян М.Н., Демидова А.В., Велиева Т.Р.* Аналитико-численная реализация алгебры поливекторов на языке Julia // Программирование. 2022. № 1. С. 54–64.
3. *Велиева Т.Р., Геворкян М.Н., Демидова А.В.* Аппарат геометрической алгебры и кватернионов в системах символьных вычислений для описания вращений в евклидовом пространстве // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 1. С. 31–42.
4. *Королькова А.В., Геворкян М.Н., Кулябов Д.С., Севастьянов Л.А.* Средства компьютерной алгебры для геометризации уравнений Максвелла // Программирование. 2023. Т. 49. № 4. С. 33–38.

5. *Kulyabov D. S.* Using two Types of Computer Algebra Systems to Solve Maxwell Optics Problems // Programming and Computer Software. 2016. V. 42. № 2. P. 77–83. arXiv: 1605.00832.
6. *Kulyabov D.S., Korolkova A.V., Sevastianov L.A. et al.* Algorithm for Lens Calculations in the Geometrized Maxwell Theory // Saratov Fall Meeting 2017: Laser Physics and Photonics XVIII; and Computational Biophysics and Analysis of Biomedical Data IV / Ed. by Vladimir L. Derbov, Dmitry E. Postnov. Vol. 10717 of Progress in Biomedical Optics and Imaging – Proceedings of SPIE. Saratov: SPIE, 2018. 4. P. 107170Y.1–6. arXiv : 1806.01643.
7. *Grassmann H.G.* Die Mechanik nach den Principien der Ausdehnungslehre // Mathematische Annalen. 1877. 6. Bd. 12, H. 2. S. 222–240.
8. *Kuipers J.B.* Quaternions And Rotation Sequences. Princeton, New Jersey: Princeton University Press, 2002. 400 p.
9. *Clifford W.K.* Applications of Grassmann's Extensive Algebra // American Journal of Mathematics. 1878. V. 1. № 4. P. 350–358.
10 GAlgebra — Symbolic Geometric Algebra/ Calculus package for SymPy. 2023. https://galgebra.readthedocs.io/en/latest/index.html.
11. *Velieva T.R., Gevorkyan M.N., Demidova A.V. et al.* Geometric Algebra and Quaternion Techniques in Computer Algebra Systems for Describing Rotations in Eucledean Space // Computational Mathematics and Mathematical Physics. 2023. V. 63. № 1. P. 29–39.
12. *Sandon D.* Symbolic Computation with Python and SymPy. 2021. V. 1. 580 p.
13. *Sandon D.* Symbolic Computation with Python and SymPy. 2021. V. 2. 429 p.
14. The international system of units (SI) / Ed. by David B. Newell, Eite Tiesinga. NIST Special Publication 330. National Institute of Standards and Technology, 2019. Aug. 122 p.
15. *Dorst L., Fontijne D., Mann S.* Geometric algebra for computer science (with errata). The Morgan Kaufmann Series in Computer Graphics. 1 edition. Morgan Kaufmann, 2007.
16. *de Sabbata V., Datta B.K.* Geometric Algebra and Applications to Physics. Taylor & Francis, 2006. 12. 184 p.
17. *Rosn A.* Geometric Multivector Analysis. Springer International Publishing, 2019. 465 p.
18. *Rodrigues Jr W.A., de Oliveira E.C.* The Many Faces of Maxwell, Dirac and Einstein Equations. Springer International Publishing, 2016. V. 922 of Lecture Notes in Physics. 587 p.
19. *Doran C., Lasenby A.* Geometric Algebra for Physicists. Cambridge University Press, 2003. May. 578 p.
20. *Chisolm E.* Geometric Algebra. 2012. arXiv : 1205.5935.
21. *Lasenby A., Doran C., Arcaute E.* Applications of Geometric Algebra in Electromagnetism, Quantum Theory and Gravity // Clifford Algebras / Ed. by R. Abamowicz. Birkhuser Boston, 2004. Vol. 34 of Progress in Mathematical Physics. 467–489 p.
22. *Toomey D.* Learning Jupyter. Packt Publishing Ltd., 2016. 305 p.

SYMBOLIC STUDIES OF MAXWELL'S EQUATIONS IN SPACE-TIME ALGEBRA FORMALISM

© 2024 г. A. V. Korol'kova[a], M. N. Gevorkyan[a], A. V. Fedorov[a], K. A. Shtepa[a], D. S. Kulyabov[a, b]

[a] *RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198 Russia*
[b] *Joint Institute for Nuclear Research, Dubna, Moscow oblast, 141980 Russia*

Different implementations of Clifford algebra: spinors, quaternions, and geometric algebra, are used to describe physical and technical systems. The geometric algebra formalism is a relatively new approach, destined to be used primarily by engineers and applied researchers. In a number of works, the authors examined the implementation of the geometric algebra formalism for computer algebra systems. In this article, the authors extend elliptic geometric algebra to hyperbolic space-time algebra. The results are illustrated by different representations of Maxwell's equations. Using a computer algebra system, Maxwell's vacuum equations in the space-time algebra representation are converted to Maxwell's equations in vector formalism. In addition to practical application, the authors would like to draw attention to the didactic significance of these studies.

Keywords: geometric algebra, Clifford algebra, computer algebra system SymPy, Galgebra

REFERENCES

1. *Gevorkyan M.N., Korol'kova A.V., Kulyabov D.S., Demidova A.V, Velieva T.R.* Implementation of geometric algebra in computer algebra systems // Program. Comput. Software, 2023, vol. 49, no. 1, pp. 42–48.
2. *Gevorkyan M.N., Demidova A.V., Velieva T.R., Korol'kova A.V., Kulyabov D.S.* Analitical-numerical implementation of polyvector algebra in Julia // Program. Comput. Software, 2022, vol. 48, no. 1, pp. 49–58.
3. *Velieva T.R., Gevorkyan M.N., Demidova A.V., Korol'kova A.V., Kulyabov D.S.* Geometric algebra

quaternion techniques in computer algebra system for describing rotation in Euclidean space, Zh. Vychisl. Mat. Mat. Fiz., 2023, vol. 63, no. 1, pp. 31–42.
4. *Korol'kova A.V., Gevorkyan M.N., Kulyabov D.S., Sevast'yanov L.A.* Computer algebra tools for geometrization of Maxwell's equations, Program. Comput. Software, 2023, vol. 49, no. 4, pp. 366–371.
5. *Kulyabov D.S.* Using two types of computer algebra systems to solve Maxwell optics problems, Program. Comput. Software, 2016, vol. 42, no. 2, pp. 77–83. arXiv: 1605.00832.
6. *Kulyabov D.S., Korolkova A.V.* Algorithm for lens calculations in the geometrized Maxwell theory, SaratovFall Meeting 2017: Laser Physics and PhotonicsXVIII; and Computational Biophysicsand Analysis of Biomedical Data IV; Proceedings of SPIE, Saratov: SPIE, 2018. arXiv: 1806.01643.
7. *Grassmann H.G.* Die mechanik nach den principiender ausdehnungslehre, Mathematische Annalen, 1877, vol. 12, no. 2, pp. 222–240.
8. *Kuipers J.B.* Quaternions and Rotation Sequences, Princeton, New Jersey: Princeton University Press, 2002.
9. *Clifford W.K.* Applications of grassmann's extensive algebra, Am. J. Math., 1878, vol. 1, no. 4, pp. 350–358.
10. GAlgebra Symbolic Geometric Algebra/Calculus Package for SymPy, 2023. URL: https://galgebra.readthedocs.io/en/latest/index.html.
11. *Velieva T.R., Gevorkyan M.N., Demidova A.V., Korol'kova A.V., Kulyabov D.S.* Geometric algebra quaternion techniques in computer algebra system for describing rotation in Euclidean space // Comput. Math. Math. Phys., 2023, vol. 63, no. 1, pp. 29–39.
12. *Sandon D.* Symbolic Computation with Pythonand SymPy, 2021, vol. 1, p. 580.
13. *Sandon D.* Symbolic Computation with Pythonand SymPy, 2021, vol. 2, p. 429.
14. The international system of units (SI), David, B., Ed., Newell: Eite Tiesinga. NIST SpecialPublication, 2019.
15. *Dorst L., Fontijne D., Mann S.* Geometric Algebra for Computer Science (with Errata). The Morgan Kaufmann Series in Computer Graphics, Morgan Kaufmann, 2007.
16. *de Sabbata V., Datta B.K.* Geometric Algebra and Applications to Physics, Taylor & Francis, 2006.
17. *Rosn A.* Geometric Multivector Analysis. Springer, 2019.
18. *Rodrigues Jr. W.A., de Oliveira E.C.* The Many-Faces of Maxwell, Dirac and Einstein Equations. Springer, 2016.
19. *Doran C., Lasenby A.* Geometric Algebra for Physicists. Cambridge University Press, 2003.
20. *Chisolm E.* Geometric Algebra, 2012. arXiv: 1205.5935
21. *Lasenby A., Doran C., Arcaute E.* Applications of geometric algebra in electromagnetism, quantum theory and gravity, in Clifford Algebras, Abamowicz, R., Ed., Boston: Birkhuser, 2004.
22. *Toomey D.* Learning Jupyter, Packt Publishing Ltd., 2016.

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 74–78*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 519.85

О ВЫЧИСЛЕНИИ ЧАСТИЧНЫХ СУММ КРАТНЫХ ЧИСЛОВЫХ РЯДОВ МЕТОДАМИ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

© 2024 г. В. И. Кузоватов[a], *, А. А. Кытманов[b, a], **, Е. К. Мышкина[a], ***

[a] *Сибирский федеральный университет, 660041 Красноярск, пр. Свободный, 79, Россия*
[b] *МИРЭА — Российский технологический университет, 119454 Москва, пр. Вернадского, 78, Россия*

**E-mail: kuzovatov@yandex.ru*
***E-mail: aakytm@gmail.com*
****E-mail: elfifenok@mail.ru*

Поступила в редакцию 01.08.2023
После доработки 10.09.2023
Принята к публикации 01.10.2023

В работе предложен метод вычисления частичных сумм некоторых кратных числовых рядов, возникающих в процессе нахождения результанта многочлена и целой функции. Степенные суммы корней, участвующие в данной формуле, возможно найти без нахождения самих корней системы с помощью символьного алгоритма, использующего рекуррентные формулы Ньютона. Алгоритм, реализующий предложенный подход вычисления частичных сумм кратных числовых рядов, реализован в системе компьютерной алгебры Maple. Приведены примеры нахождения частичных сумм некоторых классов кратных числовых рядов с помощью созданного алгоритма.

Ключевые слова: кратные числовые ряды, степенные суммы корней, рекуррентные формулы Ньютона, методы компьютерной алгебры

DOI: 10.31857/S0132347424020094 **EDN:** ROLFAU

1. ВВЕДЕНИЕ

Исследование систем алгебраических уравнений является классической задачей. Частью ее является задача исключения неизвестных. Для двух переменных и систем из двух уравнений она решается с помощью результанта Сильвестра. Для систем из большего числа уравнений хорошо известна классическая схема исключения неизвестных, но она, как правило, является весьма трудоемкой. В настоящее время общепринятым методом исключения неизвестных является метод базисов Гребнера, созданный в работах Бухбергера и его учеников.

Модифицированный метод исключения неизвестных из систем алгебраических уравнений в $\mathbb{C}^n$ возник в работе Л.А. Айзенберга [1]. Основная идея метода заключается в нахождении степенных сумм корней системы с помощью формулы многомерного логарифмического вычета, не вычисляя самих корней, а затем в использовании классических рекуррентных формул Ньютона для построения результанта. В отличие от классического метода исключения он менее трудоемок и не увеличивает кратности корней. Дальнейшая его разработка продолжена в монографиях [2, 3, 4]. В качестве приложений этой теории были рассмотрены системы нелинейных уравнений, возникающие в химической кинетике и зависящие от параметров.

Во многих прикладных задачах [5] возникают также неалгебраические системы уравнений, состоящие из экспоненциальных многочленов, т.е. из функций конечного порядка роста. Для систем неалгебраических уравнений, множество корней которых, как правило, бесконечно, степенные суммы корней в положительной степени, вообще говоря, являются расходящимися рядами. Но степенные суммы корней в отрицательной степени часто являются сходящимися. Возникает задача об их вычислении через коэффициенты Тейлора функций, входящих в систему. Это вычисление можно осуществить с помощью вычетных интегралов. Тем самым возник метод нахождения сумм кратных числовых рядов, основанный на использовании вычетных интегралов, связанных с системой уравнений. В работах [6, 7, 8] на основе данного метода найдены суммы некоторых классов кратных числовых рядов, ранее отсутствовавших в известных справочниках.

Отметим, что метод вычисления сумм кратных числовых рядов, представленный в данной статье, существенно отличается от метода вычетных интегралов, рассмотренного в работах [6, 7, 8]. Наш подход основан на использовании формулы для результанта многочлена (или целой функции с конечным числом нулей) и целой функции, полученной в работах [9, 10]. Данная формула не требует значения

корней исследуемых функций и представляет собой некоторое комбинаторное выражение. Вычисляя результант многочлена и целой функции двумя разными способами, удается получить соотношение для кратных числовых рядов. В качестве второго способа нахождения результанта выбирается формула для произведения одной функции в корнях другой. В данной статье мы приводим примеры вычисления сумм некоторых типов кратных числовых рядов, ранее отсутствовавших в известных справочниках. Они выражаются через известные специальные функции, такие как функция Бесселя. Однако, не всегда аналитически удается выразить сумму кратного числового ряда через известные величины. Для некоторых кратных числовых рядов, суммы которых могут быть вычислены предложенным методом, другие методы их вычисления на данный момент неизвестны. Описанию предложенного метода и посвящена настоящая статья.

2. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

Для данных многочленов f и g классический результант $R(f,g)$ может быть определен различными способами с использованием определителя Сильвестра, способа Безу–Кэли или формулы для произведения

$$R(f,g) = \prod_{\{x:f(x)=0\}} g(x).$$

Подробный обзор работ, относящихся к классическому результанту и его обобщениям, можно найти в [10].

Напомним классические рекуррентные формулы Ньютона для многочленов. Они связывают между собой коэффициенты многочлена и степенные суммы его корней.

Пусть

$$P(z) = z^m + c_1 z^{m-1} + \ldots + c_{m-1} z + c_m.$$

Обозначим через $z_1, z_2, \ldots, z_m$ его корни (среди них могут быть и кратные). Определим степенную сумму корней

$$S_k = z_1^k + \ldots + z_m^k, \quad k \in \mathbb{N}, \quad S_0 = m.$$

Степенные суммы S_k и коэффициенты c_j связаны между собой классическими рекуррентными формулами Ньютона:

$$\begin{cases} S_j + \sum\limits_{i=1}^{j-1} S_{j-i} c_i + j c_j = 0, & 1 \leqslant j \leqslant m, \\ S_j + \sum\limits_{i=1}^{m} S_{j-i} c_i = 0, & j > m. \end{cases}$$

В работах [11, 12] предложены метод и символьный алгоритм вычисления многомерных рекуррентных формул Ньютона.

Рассмотрим систему уравнений, состоящую из двух многочленов $f(z)$ и $g(z)$ степеней m и n соответственно:

$$\begin{cases} f(z) = a_0 + a_1 z + \ldots + a_{m-1} z^{m-1} + z^m, \\ g(z) = b_0 + b_1 z + \ldots + b_n z^n. \end{cases} \tag{1}$$

Теорема 1 ([9]). *Результант $R(f,g)$ системы многочленов вида* (1)*, где $m=2$, вычисляется по формуле*

$$R(f,g) = \sum_{k=0}^{n} b_k^2 a_0^k + \sum_{t=0}^{n-1} \sum_{s=t+1}^{n} b_t b_s a_0^t S_{s-t}, \tag{2}$$

где степенные суммы S_j корней многочлена $f(z)$ определяются рекуррентными формулами Ньютона.

Теорема 2 ([10]). *Результант $R(f,g)$ системы многочленов вида* (1)*, где $m=3$, вычисляется по формуле*

$$\begin{aligned} R(f,g) = {} & \sum_{k=0}^{n} b_k^3 (-a_0)^k + \sum_{s=0}^{n} \sum_{t=s+1}^{n} (-a_0)^s \times \\ & \times \left(\frac{1}{2} b_s b_t^2 \left(S_{t-s}^2 - S_{2t-2s} \right) + b_s^2 b_t S_{t-s} \right) + \\ & + \sum_{s=0}^{n} \sum_{t=s+1}^{n} \sum_{p=t+1}^{n} b_s b_t b_p (-a_0)^s \\ & \times \left[S_{t-s} \cdot S_{p-s} - S_{t+p-2s} \right]. \end{aligned} \tag{3}$$

Осуществив в (2) (соответственно, в (3)) предельный переход по n, получаем утверждение о результанте относительно многочлена (или целой функции с конечным числом нулей) и целой функции.

Теорема 3 ([9, 10]). *Пусть $g(z)$ — целая функция на комплексной плоскости $\mathbb{C}$ вида*

$$g(z) = b_0 + b_1 z + b_2 z^2 + \ldots + b_n z^n + \ldots,$$

а $f(z)$ — многочлен вида (1) при $m=2$ (или соответственно при $m=3$). Тогда результантом $R(f,g)$ является выражение (2) (соответственно выражение (3)), в котором необходимо выполнить предельный переход по n.

Здесь следует отметить, что ряды, получающиеся при предельном переходе по n в правых частях формул (2) и (3), абсолютно сходятся ввиду того, что b_k — это коэффициенты разложения целой функции.

Замечание 1. *Отметим, что такие задачи, как вычисление результанта двух многочленов или вычисление степенных сумм системы уравнений, тесно связаны с теорией симметрических функций. Современные системы компьютерной алгебры общего назначения,*

такие как Maplesoft Maple, Wolfram Mathematica и свободно распространяемая система SageMath, содержат пакеты, позволяющие эффективно проводить вычисления с симметрическими многочленами и на основе этого решать такие задачи, как вычисления результанта. Тем не менее ценность результатов, приведенных в теоремах и в том, что там предлагаются явные выражения для вычисления результанта через коэффициенты многочленов и их степенные суммы.

3. ОСНОВНОЙ РЕЗУЛЬТАТ

Целью данной работы является программная реализация формул (2) и (3). При определенном выборе многочлена $f(z)$ и целой функции $g(z)$ в формуле для результанта в теореме 3 мы получаем кратные числовые ряды. Задача о нахождении частичных сумм данных кратных числовых рядов есть предмет нашего исследования. Таким образом, задача о вычислении кратных числовых рядов, не возникающих при вычислении результанта в теореме 3, остается открытой задачей. Отметим, что имеется большое число способов вычисления результанта двух многочленов в различных системах компьютерной алгебры [13]. Наш подход к построению результанта позволяет находить частичные суммы рассматриваемых рядов, что является особенно важным в случае, когда корни функций не известны.

Алгоритм вычисления частичных сумм кратных числовых рядов был реализован в среде Maple 2016 64bit. Полный код программы, реализующий формулы (2) и (3), доступен по адресам https://github.com/aakytmanov/Resultant2n, https://github.com/aakytmanov/Resultant34n. Вычисления производились на машине Intel Core i7-4790 (3.6 GHz) с 32 Gb RAM под управлением Windows 10 Pro x64 22H2. Время счета для приведенных ниже примеров составило менее 1 секунды.

Пример 1. *Рассмотрим систему уравнений*

$$\begin{cases} f(z) = (z-a)(z-b) = z^2 - z(a+b) + ab, \\ g(z) = \operatorname{sh} z = \displaystyle\sum_{n=0}^{\infty} \frac{z^{2n+1}}{(2n+1)!}. \end{cases}$$

Поскольку в данной задаче в разложении целой функции $g(z)$ присутствуют лишь коэффициенты при нечетных степенях, то формула для результанта из теоремы 3 примет вид:

$$R(f,g) = \sum_{k=0}^{\infty} b_{2k+1}^2 a_0^{2k+1} + \sum_{t=0}^{\infty}\sum_{s=t+1}^{\infty} b_{2t+1} b_{2s+1} a_0^{2t+1} S_{2s+1-(2t+1)}.$$

Используя данное соотношение и определение результанта в виде формулы для произведения, получим

$$\sum_{k=0}^{\infty}\left(\frac{1}{(2k+1)!}\right)^2 (ab)^{2k+1} + \sum_{t=0}^{\infty}\sum_{s=t+1}^{\infty} \frac{1}{(2t+1)!}\frac{1}{(2s+1)!}(ab)^{2t+1} S_{2s-2t} = \operatorname{sh} a \operatorname{sh} b.$$

Здесь

$$S_{2s-2t} = z_1^{2s-2t} + z_2^{2s-2t} = a^{2s-2t} + b^{2s-2t},$$

$$(ab)^{2t+1} S_{2s-2t} = a^{2s+1} b^{2t+1} + a^{2t+1} b^{2s+1}.$$

Таким образом,

$$\sum_{k=0}^{\infty} \frac{(ab)^{2k+1}}{\left[(2k+1)!\right]^2} + \sum_{t=0}^{\infty}\sum_{s=t+1}^{\infty} \frac{a^{2s+1}b^{2t+1} + a^{2t+1}b^{2s+1}}{(2t+1)!(2s+1)!} = \operatorname{sh} a \operatorname{sh} b. \quad (4)$$

Известно [14, формула 5.2.10.5], что

$$\sum_{k=0}^{\infty} \frac{(ab)^{2k+1}}{\left[(2k+1)!\right]^2} = \frac{1}{2}\left[I_0\left(2\sqrt{ab}\right) - J_0\left(2\sqrt{ab}\right)\right],$$

где $J_0(2\sqrt{ab})$ — функция Бесселя 1-го рода и $I_0(2\sqrt{ab})$ — модифицированная функция Бесселя 1-го рода (функция Бесселя мнимого аргумента). Следовательно, мы можем выразить сумму кратного числового ряда следуюшим образом:

$$\sum_{t=0}^{\infty}\sum_{s=t+1}^{\infty} \frac{a^{2s+1}b^{2t+1} + a^{2t+1}b^{2s+1}}{(2t+1)!(2s+1)!} = \operatorname{sh} a \operatorname{sh} b - \frac{1}{2}\left[I_0(2\sqrt{ab}) - J_0(2\sqrt{ab})\right].$$

Частичные суммы выражения в левой части равенства (4) могут быть получены с помощью предложенного алгоритма. Например, используя входные данные

```
> simplify(Res([a*b, -a-b], [0, 1, 0, 1/3!,
0, 1/5!, 0, 1/7!]));
```

получим

$$\frac{ab\left(b^6 + 42b^4 + 840b^2 + 5040\right)}{25\,401\,600} \times \left(a^6 + 42a^4 + 840a^2 + 5040\right).$$

Пример 2. Рассмотрим систему уравнений

$$\begin{cases} f(z) = z^3 - a^3, \\ g(z) = e^{bz} = 1 + bz + \dfrac{(bz)^2}{2!} + \ldots + \dfrac{(bz)^n}{n!} + \ldots. \end{cases}$$

Если z_1, z_2, z_3 — нули многочлена $f(z)$, то, используя определение результанта в виде формулы для произведения, получим

$$R(f,g) = g(z_1)g(z_2)g(z_3) = e^{b(z_1+z_2+z_3)}.$$

Применяя формулы Виета для кубического многочлена $f(z)$, получим

$$a_2 = -(z_1 + z_2 + z_3) = 0.$$

Таким образом, $R(f,g) = 1$. Вычисляя теперь результант $R(f,g)$, используя теорему 3, получим соотношение (см. [10])

$$\sum_{k=0}^{\infty} \frac{(ab)^{3k}}{(k!)^3} + \sum_{s=0}^{\infty}\sum_{j=1}^{\infty} a^{3s} \times$$
$$\times\left(3\frac{b^{3s+6j}}{s!(s+3j)!^2}a^{6j} + 3\frac{b^{3s+3j}}{s!^2(s+3j)!}a^{3j}\right) +$$
$$+ \sum_{s=0}^{\infty}\sum_{t=s+1}^{\infty}\sum_{p=t+1}^{\infty} b_s b_t b_p a^{3s} \times$$
$$\times\left[S_{t-s}\cdot S_{p-s} - S_{t+p-2s}\right] = 1.$$

В данном примере ни один из кратных числовых рядов не может быть выражен через известные величины. Однако, используя предложенный алгоритм, частичные суммы выражения в левой части могут быть найдены. Например, используя входные данные

```
> simplify(Res3d([-a^3, 0, 0], [1, b,
b^2/2!, b^3/3!, b^4/4!, b^5/5!, b^6/6!]));
```

получим

$$1 - \frac{a^9b^9}{4320} + \frac{a^{12}b^{12}}{345\,600} + \frac{a^{15}b^{15}}{10\,368\,000} + \frac{a^{18}b^{18}}{373\,248\,000}.$$

ЗАКЛЮЧЕНИЕ

В работе предложен компьютерно-алгебраический подход к вычислению частичных сумм кратных числовых рядов, возникающих при вычислении результанта многочлена и целой функции. Разработаны и программно реализованы символьные алгоритмы, один из которых вычисляет степенные суммы корней по рекуррентным формулам Ньютона, а второй на основе вычисления результанта позволяет находить частичные суммы некоторых классов кратных числовых рядов. Рассмотрены примеры вычисления таких рядов.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Исследования, представленные в работе, были выполнены при поддержке следующих фондов: первый и третий авторы использовали финансовую поддержку РНФ, Правительства Красноярского края и Красноярского краевого фонда науки, грант 22-21-20028.

СПИСОК ЛИТЕРАТУРЫ

1. *Aizenberg L.A.* On one formula of generalized logarithmic residue and solution of the system of nonlinear equations, Dokl. Akad. Nauk SSSR (Proc. of the USSR Academy of Sciences). 1977. V. 234. № 3. P. 505–508.
2. *Aizenberg L.A., Yuzhakov A.P.* Integral'nye predstavleniya i vychety v mnogomernom kompleksnom analize (Integral Representations and Residues in Multidimensional Complex Analysis), Novosibirsk: Nauka, 1979.
3. *Bykov V.I., Kytmanov A.M., Lazman M.Z.* Elimination methods in polynomial computer algebra, Kluwer Academic Publishers, Dodrecht-Boston-Basel, 1998.
4. *Tsikh A.K.* Multidimensional residues and their applications, AMS, Providence, 1992.
5. *Bykov V.I., Tsybenova S.B.* Nelineinye modeli khimicheskoi kinetiki (Nonlinear Models of Chemical Kinetics), Moscow: KRASAND, 2011.
6. *Myshkina E.K.* Some examples of finding the sums of multiple series // J. Siberian Federal Univ. Math. Phys. 2014. V. 7. № 4. P. 515–529.
7. *Kytmanov A.A., Kytmanov A.M., Myshkina E.K.* Finding Residue Integrals for Systems of Non-algebraic Equations in $\mathbb{C}^n$ // Journal of Symbolic Computation. 2015. V. 66. P. 98–110.
8. *Kytmanov A.A., Kytmanov A.M., Myshkina E.K.* Residue integrals and Waring's formulas for algebraic or even transcendental systems // Complex Variables and Elliptic Equations. 2019. V. 64. № 1. P. 93–111.
9. *Kytmanov A.M., Myshkina E.K.* On Some Approach for Finding the Resultant of Two Entire Functions // J. Siberian Federal Univ. Math. Phys. 2019. V. 12. № 4. P. 434–438.
10. *Kytmanov A.M., Myshkina E.K.* On Finding the Resultant of Two Entire Functions // Probl. Anal. Issues Anal. 2020. V. 9. № 3. P. 119–130.
11. *Kytmanov A.A.* Analogues of recurrent Newton formulas, Russian Math. 2009. V. 53. P. 34–44.
12. *Kytmanov A.A.* An algorithm for calculating power sims of roots for a class of systems of nonlinear equations // Programming and Computer Software. 2010. V. 36. № 2. P. 103–110.
13. *Bryuno A.D., Batkhin A.B.* Algorithms and programs for calculating the roots of polynomial of one or two variables // Programming and Computer Software. 2021. V. 47. № 5. P. 353–373.
14. *Prudnikov A.P., Brychkov Yu.A., Marichev O.I.* Integraly i ryady. Elementarnye funktsii (Integrals and Series. Elementary Functions), Moscow: Nauka, Gl. red. fiz.-mat. lit, 1981.

ON CALCULATING PARTIAL SUMS OF MULTIPLE NUMERICAL SERIES BY METHODS OF COMPUTER ALGEBRA

© 2024 V. I. Kuzovatov[a], A. A. Kytmanov[a, b], E. K. Myshkina[a]

[a] *Siberian Federal University, Krasnoyarsk, Russia*
[b] *MIREA — Russian Technological University, Moscow, Russia*

A method to calculate partial sums of some multiple numerical series arising when searching for the resultant of a polynomial and an entire function is proposed. One can apply a symbolic algorithm that uses recurrent Newton formulas to find power sums of roots included in this formula without finding the very roots of the system. The algorithm that implements the proposed approach to calculate partial sums of multiple numerical series is implemented in Maple. Examples of using this algorithm to find partial sums of some classes of multiple numerical series are given.

Keywords: multiple numerical series, power sums of roots, recurrent Newton formulas, methods of Computer Algebra

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 79–83*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6 + 517.55

ПРИМЕНЕНИЕ СИСТЕМ КОМПЬЮТЕРНОЙ АЛГЕБРЫ ДЛЯ ИССЛЕДОВАНИЯ ТОЖДЕСТВ ЧАУНДИ-БУЛЛАРДА ДЛЯ ФУНКЦИИ ВЕКТОРНОГО РАЗБИЕНИЯ С ВЕСОМ

© 2024 г. А. Б. Лейнартене[a], *, А. П. Ляпин[a], **

[a]*Сибирский федеральный университет*
660041 Красноярск, пр. Свободный, д. 79, Россия

**E-mail: aleina@mail.ru*
***E-mail: aplyapin@sfu-kras.ru*

Поступила в редакцию 28.06.2023
После доработки 10.08.2023
Принята к публикации 01.10.2023

В данной работе предложен алгоритм получения тождества Чаунди–Булларда для функции векторного разбиения с весом с использованием методов компьютерной алгебры. Для автоматизации данного процесса в среде Maple был разработан и реализован алгоритм, вычисляющий значения функции векторного разбиения с весом путем нахождения неотрицательных решений систем линейных диофантовых уравнений, на основе которых происходит составление указанных тождеств. Входными данными алгоритма является набор целочисленных векторов, образующих заостренный решеточный конус, и некоторая точка из данного конуса, выходными данными – тождество Чаунди–Булларда для функции векторного разбиения с весом. Указанный код размещен в депозитории и готов к использованию. Приведен пример, демонстрирующий работу данного алгоритма.

Ключевые слова: тождество Чаунди–Булларда, функция векторного разбиения, решеточный конус
DOI: 10.31857/S0132347424020105 **EDN:** ROHMGQ

1. ВВЕДЕНИЕ И ПОСТАНОВКА ЗАДАЧИ

Методы компьютерной алгебры показали свою эффективность в исследовании широкого класса задач из теории многомерных разностных уравнений (см., например, [1]–[7]). В данной работе предложен алгоритм получения тождеств Чаунди–Булларда для функции векторного разбиения с весом с использованием системы компьютерной алгебры Maple.

Для целых неотрицательных чисел m,n обозначим

$$P_{m,n}(x) = (1-x)^{n+1}\sum_{k=0}^{m}\binom{n+k}{k}x^k.$$

Элегантное соотношение

$$P_{m,n}(x) + P_{n,m}(1-x) = 1, \qquad (1.1)$$

известное как тождество Чаунди–Булларда, было получено в работе [8] в 1960 г.

Подробный обзор тождества Чаунди–Булларда и различных подходов к его доказательству приведен в [9]. В [10] подобные тождества были получены с использованием метода производящих функций и свойств композиции Адамара кратных степенных рядов (подробнее о многомерном аналоге композиции Адамара см. [11]). В [12] отмечалось, что Монмор получил это тождество в 1713 г., затем Муавр обнаружил его в 1738 г. и Геринг применял его в 1868 г. Тождество Чаунди–Булларда появляется в теории рекурсивых цифровых фильтров (см. [13]), в теории вейвлетов (см. [14]), в теории гипергеометрических функций Гаусса (см. [15]). В работе [16] соотношение (1.1) называется тождеством Добеши в случае $m=n$.

В настоящее время появилось немало исследований по данной тематике. Например, в 2023 г. в работе [17] несколько тождеств с использованием символа Похгаммера были доказаны при помощи тождества Чаунди–Булларда. В 2016 г. в работе [18] были представлены два новых доказательства данного тождества на основе дифференцирования суммы ряда бесконечной геометрической прогрессии в связи с исследованием комбинаторной задачи о справедливом разделе ставки. В 2014 г. в работе [19] были представлены однородная форма тождества Чаунди–Булларда и ее очевидное обобщение на случай n переменных. Некоторые полезные соотношения, связанные с тождеством Чаунди–Булларда, приведены в работе [20].

Обозначим $\mathbb{Z},\mathbb{Z}_{\geqslant},\mathbb{C}$ — множества целых, целых неотрицательных и комплексных чисел соответственно, $\mathbb{Z}^n = \mathbb{Z}\times\ldots\times\mathbb{Z}$, $\mathbb{Z}^n_{\geqslant} = \mathbb{Z}_{\geqslant}\times\cdots\times\mathbb{Z}_{\geqslant}$.

Функция векторного разбиения $P_\Delta(\lambda)$ является числом представлений вектора λ через набор заданных векторов $\Delta = \{\alpha^1,\ldots,\alpha^N\} \subset \mathbb{Z}^n$ с целыми неотрицательными коэффициентами. Данная функция изучалась в [21] и [22] в связи с исследованиями рациональных многогранников. Функцию векторного разбиения можно также рассматривать как число целых неотрицательных решений диофантова уравнения $Ax = \lambda$, где A – матрица, стобцы которой являются векторами из набора Δ (см. [23]):

$$P_\Delta(\lambda) = \sum_{Ax=\lambda, x\in\mathbb{Z}^n_{\geqslant}} 1.$$

Аналоги функции векторного разбиения в целочисленном конусе исследовались в [24] и [25] в связи с обобщением теоремы Римана–Роха и теории индексов трансверсальных эллиптических операторов. Структурная теорема для функции векторного разбиения и эффективные инстументы для ее вычисления были даны в работе [26].

Пусть $\varphi : \mathbb{Z}^n_{\geqslant} \to \mathbb{C}$. В работе [27] была определена функция векторного разбиения с весом φ

$$P_\Delta(\lambda;\varphi) = \sum_{Ax=\lambda, x\in\mathbb{Z}^n_{\geqslant}} \varphi(x)$$

и доказано тождество Чаунди–Булларда для функции векторного разбиения с весом. Отметим, что в данной работе были получены многомерные разностные уравнения, решения которых являются функциями векторного разбиения с весом $P_\Delta(\lambda;\varphi)$, и найдены их производящие функции (подробнее см. [28], [29], [30]).

2. ТОЖДЕСТВО ЧАУНДИ–БУЛЛАРДА

Пусть $\Delta = \{\alpha^1,\ldots,\alpha^N\} \subset \mathbb{Z}^n$ и решеточный конус

$$K = \left\{ \begin{matrix} \lambda \in \mathbb{Z}^n : \lambda = x_1\alpha^1 + \cdots + x_N\alpha^N, \\ x_1,\ldots,x_N \in \mathbb{Z}_{\geqslant} \end{matrix} \right\}$$

– заостренный, то есть такой, для которого выполняется условие $\lambda \in K$ и $-\lambda \in K$, если и только если $\lambda = 0$. Пусть $A = [\alpha^1,\ldots,\alpha^N]$ — матрица из n строк и N столбцов, составленная из векторов-столбцов $\{\alpha^1,\ldots,\alpha^N\}$. Для произвольной функции целочисленных аргументов $\varphi : \mathbb{Z}^N_{\geqslant} \to \mathbb{C}$ определим функцию векторного разбиения с весом $\varphi(x)$ следующим образом:

$$P_A(\lambda;\varphi) = \sum_{\substack{x: Ax=\lambda \\ x\in\mathbb{Z}^N_{\geqslant}}} \varphi(x), \lambda \in \mathbb{Z}^n.$$

При $\varphi(x) \equiv 1$ данная функция совпадает с классической функцией векторного разбиения P_A.

Обозначим наборы вектор-столбцов $\Delta_j = \Delta\backslash\{\alpha^j\}$, $j = 1,\ldots,N$ матрицы $A_j, j = 1,\ldots,N$, составленные из таких наборов соответственно:

$$A_j = [\alpha^1,\ldots,\alpha^{j-1},\alpha^{j+1},\ldots,\alpha^N]$$

и заостренные решеточные конусы

$$K_j = \left\{ \begin{matrix} \nu \in \mathbb{Z}^n : \nu = y_1\alpha^1 + \ldots[j]\ldots + \\ +y_N\alpha^N, y_1,\ldots,[j],\ldots,y_N \in \mathbb{Z}_{\geqslant} \end{matrix} \right\}.$$

Также обозначим $c^x = c_1^{x_1}\cdots c_N^{x_N}$, $c^{x+e^j} = c_1^{x_1}\cdots c_j^{x_j+1}\cdots c_N^{x_N}$, $|x| = x_1 + \cdots + x_n$.

Тогда справедливо следующее утверждение.

Теорема 1. *Если* $c_1 + c_2 + \cdots + c_N = 1$ *и* $\varphi_j(x) = \frac{|x|!}{x!}c^{x+e^j}$, *тогда для любого* $\mu = (\mu_1,\ldots,\mu_N) \in \mathbb{Z}^N$ *выполняется тождество*

$$\sum_{j=1}^{N}\sum_{\nu\in K_j} P_{A_j}(\nu)P_A(\mu - \nu;\varphi_j) = P_A(\mu). \qquad (2.1)$$

Доказательство данной теоремы приведено в работе [27].

3. ОПИСАНИЕ АЛГОРИТМА

Рассмотрим алгоритм получения тождеств Чаунди–Булларда при помощи решения систем линейных диофантовых уравнений вида $Ax = \lambda$ и вычисления функций векторного разбиения с весом $P_A(\lambda;\varphi)$ в системе компьютеной алгебры.

Входными данными для алгоритма являются:

1. Матрица A, составленая из вектор-столбцов из набора $\Delta = \{\alpha^1,\ldots,\alpha^N\}$.

2. Некоторая точка $\lambda \in K$, где $K = \langle\alpha^1,\ldots,\alpha^N\rangle$ — конус, образованный векторами из Δ.

Выходными данными алгоритма является тождество Чаунди–Булларда для функции векторного разбиения с весом.

Описание входных и выходных данных и работы алгоритма:

1. Строим набор векторов $\Delta = \{\alpha^1,\ldots,\alpha^N\}$ из столбцов заданной матрицы A.

2. Строим конус $\lambda - K$, где K – конус, натянутый на вектора из набора Δ (строим достаточное количество точек конуса, что обеспечивается выбором достаточно большого значения параметра *interval*).

3. Строим множества точек L_j, образованные пересечением конуса $\lambda - K$ с конусами K_j, натянутыми на вектора из набора $\Delta \setminus \{\alpha^j\}$.

4. Для каждого множества $L_j, j = 1,\ldots,N$ строим соответствующие группы слагаемых, домноженных на c_j, для их вычисления используется функция $\mathrm{VPF}(A,\lambda,j,interval,\varphi)$ – сокращение от Vector Partition Function. Для каждой точки $\lambda \in L_j$ функция действует следующим образом: если $j>0$, то функция возвращает многочлен $P_{A_j}(\lambda,\varphi)$; если $j=0$, то функция возвращает $P_A(\lambda;\varphi)$ – значение функции векторного разбиения с весом $\varphi(x)$.

5. Объединяем полученные группы слагаемых в общую сумму *CBI* – сокращение от Chaundy and Bullard Indentity, которая и будет левой частью тождества Чаунди–Булларда для функции векторного разбиения с весом; если $\varphi(x)=1$, то функция возвращает значение «класссической» функции векторного разбиения.

6. Вычисляем правую часть тожества, используя $\mathrm{VPF}(A,\lambda,0,interval,1)$.

Алгоритм был реализован в среде Maple 18. Полный код программы доступен по ссылке https://github.com/lyapinap/ALeinartene2023. Вычисления производились на машине Intel(R) Core(TM) i5-1135G7 CPU 2.40 GHz, 64bit, ОЗУ 16.00 Гб под управлением Windows 11.

4. ПРИМЕР

Пусть $n=2$ и $N=3$. Рассмотрим набор векторов

$$\alpha^1 = \begin{pmatrix}1\\0\end{pmatrix}, \alpha^2 = \begin{pmatrix}1\\1\end{pmatrix}, \alpha^3 = \begin{pmatrix}0\\1\end{pmatrix} \text{ и } \lambda = \begin{pmatrix}3\\1\end{pmatrix},$$

тогда $\Delta = \{\alpha^1,\alpha^2,\alpha^3\}$, $\Delta_1 = \{\alpha^2,\alpha^3\}$, $\Delta_2 = \{\alpha^1,\alpha^3\}$, $\Delta_3 = \{\alpha^1,\alpha^2\}$, A, A_1, A_2, A_3 — матрицы, составленные из данных векторов соответственно, и K, K_1, K_2, K_3 — целочисленные конусы, натянутые на вектора из данных наборов соответственно.

Тогда тождество Чаунди–Булларда имеет вид:

$$\sum_{\nu\in K_1} P_{A_1}(\nu)P_A(\lambda-\nu;\varphi_1) + \sum_{\nu\in K_2} P_{A_2}(\nu)P_A(\lambda-\nu;\varphi_2) + \sum_{\nu\in K_3} P_{A_3}(\nu)P_A(\lambda-\nu;\varphi_3) = P_A(\lambda).$$

Решая соответствующие системы линейных диофантовых уравнений, получим множества точек, связанные с каждым слагаемым (см. рис. 1–3):

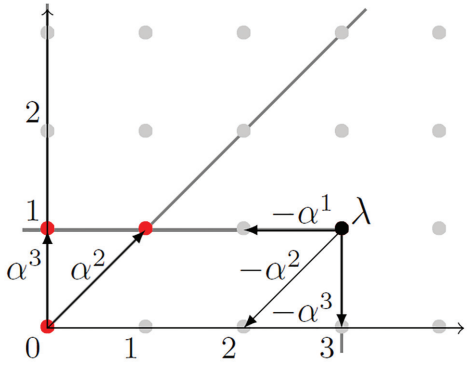


Рис. 1. Пересечение решеточных конусов $K \cap K_1$.

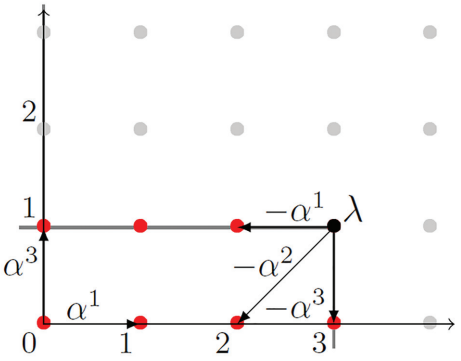


Рис. 2. Пересечение решеточных конусов $K \cap K_2$.

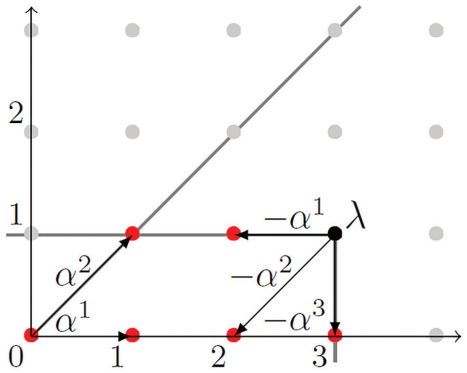


Рис. 3. Пересечение решеточных конусов $K \cap K_3$.

$$K \cap K_1 = \left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}1\\1\end{pmatrix}\right\},$$

$$K \cap K_2 = \left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\1\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix},\begin{pmatrix}3\\0\end{pmatrix},\begin{pmatrix}3\\1\end{pmatrix}\right\},$$

$$K \cap K_3 = \left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\0\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix},\begin{pmatrix}3\\0\end{pmatrix},\begin{pmatrix}3\\1\end{pmatrix}\right\}.$$

Так как $P_A(\lambda)=2$, находим тождество Чаунди–Булларда:

$$c_1\left(4c_1^3c_3+c_1^3+3c_1^2c_2+c_1^2\right)+$$
$$+c_2\begin{pmatrix}4c_1^3c_3+c_1^3+3c_1^2c_2+3c_1^2c_3+\\+c_1^2+2c_1c_2+2c_1c_3+c_1+c_2+c_3+1\end{pmatrix}+$$
$$+c_3\begin{pmatrix}4c_1^3c_3+3c_1^2c_2+3c_1^2c_3+c_1^2+\\+2c_1c_2+2c_1c_3+c_1+c_2+c_3+1\end{pmatrix}=2,$$

которое выполняется при условии, что $c_1+c_2+c_3=1$.

Для получения данного тождества с использованием разработанного алгоритма надо задать матрицу

$$A=\begin{pmatrix}1&1&0\\0&1&1\end{pmatrix},$$

точку $\lambda=(3,1)$ и, например, *interval*=5. Выполнение команды

ChaundyBullard(*A*, *lambda*, *interval*)

и дает указанный результат.

БЛАГОДАРНОСТИ

Работа поддержана Красноярским математическим центром, финансируемым Минобрнауки РФ в рамках мероприятий по созданию и развитию региональных НОМЦ (соглашение 075-02-2024-1429).

СПИСОК ЛИТЕРАТУРЫ

1. *Abramov S.A., Barkatou M.A., van Hoeij M., Petkovsek M.* Subanalytic Solutions of Linear Difference Equations and Multidimensional Hypergeometric Sequences // Journal of Symbolic Computation. 2011. № 46(11). P. 1205–1228.
2. *Abramov S.A., Petkovšek M., Ryabenko A.A.* Hypergeometric Solutions of First-order Linear Difference Dystems with Rational-function Coefficients // Lecture Notes in Computer Science. 2015. № 9301. P. 1–14.
3. *Abramov S.A., Barkatou M.A., Petkovšek M.* Linear difference operators with coefficients in the form of infinite sequences // Comput. Math. Math. Phys. 2021. № 61(10). P. 1582–1589.
4. *Abramov S.A., Ryabenko A.A., Khmelnov D.E.* Regular solutions of linear ordinary differential equations and truncated series // Comput. Math. Math. Phys. 2020. № 60(1). P. 1–14.
5. *Kytmanov A.A., Lyapin A.P., Sadykov T.M.* Evaluating the Rational Generating Function for the Solution of the Cauchy Problem for a Two-dimensional Difference Equation with Constant Coefficients // Programming and computer software. 2017. V. 43. № 2. P. 105–111.
6. *Kruchinin D., Kruchinin V., Shablya Y.* Method for Obtaining Coefficients of Powers of Multivariate Generating Functions // Mathematics, 2023. № 11. P. 2859.
7. *Chandragiri S.* Counting Lattice Paths by Using Difference Equations with Non-constant Coefficients // The Bulletin of Irkutsk State University. Series Mathematics. 2023. № 44. P. 55–70.
8. *Chaundy T.W., Bullard J.E.* John Smith's problem // Math. Gazette. 1960. V. 44. P. 253–260.
9. *Koornwinder T.H., Schlosser M.J.* On an identity by Chaundy and Bullard. I // Indag. Math.(N.S.). 2008. № 19. P. 239–261.
10. *Krivokolesko V.P., Leinartas E.K.* On identities with polynomial coefficients // Irkutsk Gos. Univ. Mat. 2012. № 5(3). P. 56–63 (in Russian).
11. *Leinartas E.K.* Multidimensional Hadamard Composition And Sums With Linear Constraints On The Summation Indices // Sib. Math. J. 1989. № 30. P. 250–255.
12. *Koornwinder T.H., Schlosser M.J.* On an identity by Chaundy and Bullard. II. More history // Indag. Math. (N.S.). 2013. № 24. P. 174–180.
13. *Herrmann O.* On the approximation problem in nonrecursive digital filter design // IEEE Trans. Circuit Theory. 1971. V. 18. P. 411–413.
14. *Daubechies I.* Ten Lectures on Wavelets, SIAM, Philadelphia, PA, 1992.
15. *Vidunas R.* Degenerate Gauss hypergeometric functions // Kyushu J. Math. № 61. 2007. P. 109–135.
16. *Zeilberger D.* On an Identity of Daubechies // Amer. Math. Monthly. 1993. № 100. P. 487.
17. *Kouba O.* A Chaundy-Bullard type identity involving the Pochhammer symbol // Indag. Math. New ser. 2023. № 34(1). P. 186–189.
18. *Zhang H.* New proofs of Chaundy-Bullard identity in "the problem of points" // Math. Intell. 2016. № 38(1). P. 4–5.
19. *Aharonov D., Elias U.* More on the identity of Chaundy and Bullard // J. Math. Anal. Appl. 2014. № 419(1). P. 422–427.
20. *Alzer H.* On a combinatorial sum // Indag. Math. New Ser. 2015. № 26(3). P. 519–525.
21. *Brion M., Vergne M.* Residue formulae, vector partition functions and lattice points in rational polytopes // J. American Math. Soc. 1997. V. 10. № 4. P. 797–833.
22. *Beck M., Gunnells P.E., Materov E.* Weighted lattice point sums in lattice polytopes, unifying Dehn-Sommerville and Ehrhart-Macdonald // Discrete Comput. Geom. 2021. № 65(2). P. 365–384.
23. *Stanley R.* Enumerative Combinatorics, V. 1. 1990.
24. *Pukhlikov A.V., Khovanskii A.G.* The Riemann-Roch theorem for integrals and sums of quasipolynomials on virtual polytopes // St. Petersburg Mathematical Journal. 1993. № 4. P. 789–812.

25. *De Concini C., Procesi C., Vergne M.* Vector partition functions and generalized Dahmen and Micchelli spaces // Transform. Groups. 2010. № 15(4). P. 751–773.
26. *Sturmfels B.* On vector partition functions // Journal of Combinatorial Theory. Series A. 1995. № 72. P. 302–309.
27. *Lyapin A.P., Chandragiri S.* Generating Functions For Vector Partition Functions And A Basic Recurrence Relation // Journal of Difference Equations & Applications. 2019. № 25(7). P. 1052–1061.
28. *Leinartas E.K., Nekrasova T.I.* Constant Coefficient Linear Difference Equations On The Rational Cones Of The Integer Lattice // Siberian Math. J. 2016. № 57(1). P. 74–85.
29. *Lyapin A.P., Cuchta T.* Sections of the generating series of a solution to the multidimensional difference equation // Bulletin of Irkutsk State University-Series mathematics. 2022. №. 42. P. 75–89
30. *Leinartas E.K.* Multiple Laurent Series And Difference Equations // Siberian Mathematical Journal. 2004. № 45(2). P. 321–326.

APPLYING COMPUTER ALGEBRA SYSTEMS TO STUDY CHAUNDY-BULLARD IDENTITIES FOR THE VECTOR PARTITION FUNCTION WITH WEIGHT

© 2024 A. B. Leinartene[a], A. P. Lyapin[a]

[a]*Siberian Federal University, pr. Svobodny 79, Krasnoyarsk, 660041 Russia*

An algorithm for obtaining the Chaundy-Bullard identity for a vector partition function with weight that uses computer algebra methods is proposed. To automate this process in Maple, an algorithm was developed and implemented that calculates the values of the vector partition function with weight by finding non-negative solutions of systems of linear Diophantine equations that are used to form the identities involved. The algorithm's input data is represented by the set of integer vectors that form a pointed lattice cone and by some point from this cone, and the Chaundy-Bullard identity for the vector partition function with weight is its output. The code involved is stored in the depository and is ready-to-use. An example demonstrating the algorithm's operation is given.

Keywords: Chaundy-Bullard identitiy, vector partition function, lattice cone

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 84–92*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 512.554.1

ПРИМИТИВНЫЕ ЭЛЕМЕНТЫ СВОБОДНЫХ НЕАССОЦИАТИВНЫХ АЛГЕБР НАД КОНЕЧНЫМИ ПОЛЯМИ

© 2024 г. М. В. Майсурадзе[a], *, А. А. Михалёв[a], **

[a]*Московский государственный университет имени М. В. Ломоносова, механико-математический факультет 119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия*

**E-mail: maisuradzemv@my.msu.ru*
***E-mail: aamikhalev@mail.ru*

Поступила в редакцию 01.07.2023
После доработки 10.08.2023
Принята к публикации 01.10.2023

Определено представление элементов свободных неассоциативных алгебр в виде набора многомерных таблиц коэффициентов. Рассмотрена операция нахождения частных производных элементов свободных неассоциативных алгебр в таком же виде. С помощью этого представления получен критерий примитивности элементов длины два и три в терминах рангов матриц, а также признак примитивности элементов произвольной длины. Полученный признак позволил оценить число примитивных элементов свободных неассоциативных алгебр над конечным полем с двумя образующими. Построенное представление позволяет оптимизировать алгоритмы символьных вычислений с примитивными элементами. С помощью этих алгоритмов найдено число примитивных элементов длины 4 свободной неассоциативной алгебры ранга 2 над конечным полем.

Ключевые слова: шрайерово многообразие линейных алгебр, свободные неассоциативные алгебры, примитивные элементы свободных алгебр, свободное дифференциальное исчисление в свободных алгебрах

DOI: 10.31857/S0132347424020115 **EDN:** RODPXT

1. ВВЕДЕНИЕ

Многообразие всех алгебр над полем является шрайеровым многообразием алгебр. Свободная алгебра в этом многообразии – свободная неассоциативная алгебра. Утверждение, аналогичное теореме Нильсена–Шрайера в 1947 году доказал А. Г. Курош для неассоциативных свободных алгебр: *всякая подалгебра неассоциативной свободной алгебры с любым множеством свободных образующих, отличная от нуля, является свободной*[2].

Для обозначения символов-букв будем использовать множество $X = \{x_1, x_2, \ldots, x_n\}$ в случае их большого числа либо $\{x, y, z\}$ иначе. Словом длины (степени) k будем называть упорядоченную последовательность из k символов $\chi_1 \ldots \chi_k$, $\chi_i \in X$ с заданной расстановкой скобок. Множество всех слов $W = \Gamma(X)$ образует свободный группоид без единичного элемента в алфавите X с операцией *: $x_i * x_j = x_i x_j$, $x_i * a = x_i(a)$, $a * x_i = (a)x_i$ для $x_i \in X$ и слов $a \in W$ длины больше 1, $a * b = (a)(b)$, для слов $a, b \in W$ степени больше 1.

Неассоциативной свободной алгеброй A (или короче *свободной алгеброй*) над полем F с *системой свободных образующих X* называется алгебра над F, линейной базой которой служит множество всех возможных слов относительно символов из X, при этом умножение индуцируется умножением в $\Gamma(X)$. Всякий элемент свободной алгебры, отличный от нуля, однозначно представим в виде суммы конечного числа различных слов (называемых *членами* этого элемента), взятых с отличными от нуля коэффициентами из поля F. Умножение элемента свободной алгебры на некоторый элемент α поля F сводится к умножению на α коэффициентов всех членов элемента.

Свободная алгебра с точностью до изоморфизма определяется числом свободных образующих X (мощностью X). Система элементов свободной алгебры называется *примитивной*, если ее можно дополнить до множества свободных образующих этой алгебры. Другими словами, подмножество M ненулевых элементов свободной алгебры A шраерова многообразия называется *примитивной системой элементов*, если существует множество свободных образующих алгебры A, содержащее подмножество M. Сами элементы такой системы называются *примитивными элементами*.

В начале 2000-х был получен критерий примитивности системы и отдельного элемента [7], [8, 12.5.1], см. также [1], [2]:

Система $a_1, a_2, \ldots, a_r$ элементов свободной неассоциативной алгебры A примитивна тогда и только тогда, когда матрица $(\partial(a_1), \ldots, \partial(a_r))$ обратима слева над алгеброй $U(A)$. В частности, элемент $a \in A$ является примитивным тогда и только тогда, когда существуют такие элементы $m_1, \ldots, m_n \in U(A)$, что

$$\sum\nolimits_{i=1}^{n} m_i \frac{\partial a}{\partial x_i} = 1.$$

Здесь $U(A)$ – *универсальная мультипликативная обертывающая алгебра* для алгебры A, являющаяся свободной ассоциативной алгеброй с множеством свободных образующих $S = \{r_w, l_w \mid w \in W\}$ – операторов левого и правого умножения на слова из W.

Пусть I_A – свободный правый $U(A)$-модуль с базисом $y_1, \ldots, y_n$.

$$I_A = y_1 U(A) \oplus \ldots \oplus y_n U(A).$$

Линейное отображение $\mathcal{D}\colon A \to I_A$, заданное формулами

$$\mathcal{D}(x_i) = y_i, i = 1 \ldots n$$

$$\mathcal{D}(ab) = \mathcal{D}(a)b + a\mathcal{D}(b),$$

где $a, b \in A$, является универсальным дифференцированием алгебры A. Частные производные

$$\frac{\partial}{\partial x_i}$$

элемента $f \in A$ однозначно определяются соотношением

$$\mathcal{D}(f) = \sum_{i=1}^{n} y_i \frac{\partial f}{\partial x_i}.$$

2. ВЕКТОРНЫЕ ПОДПРОСТРАНСТВА

Элементы свободной алгебры над полем F можно представлять как элементы арифметического векторного пространства – кортежи, составленные из коэффициентов членов. Можно рассматривать свободную алгебру как прямую сумму ее подпространств. При этом можно использовать различные разложения на прямые слагаемые в зависимости от задачи.

Для начала рассмотрим разложение по длине слова и расстановке скобок.

Пусть A – свободная алгебра с n образующими. Разобьем базис векторного пространства на группы по длине слова. В неассоциативном случае также потребуется дополнительное разбиение на группы по расстановке скобок. В каждой группе слов длины k будет n^k элементов, которые можно записать в виде k-мерной таблицы. Коэффициенты при этих мономах также удобно записывать в виде k-мерной таблицы.

Примеры:

• Мономы длины 0 (элементы поля в алгебрах с 1). Таблица $n^0 = 1$ коэффициентов записывается одним числом.

• x_i – мономы длины 1. Таблица $n^1 = n$ коэффициентов записывается вектором длины n.

• $x_i x_j$ – мономы длины 2. Таблица n^2 коэффициентов записывается квадратной матрицей $n \times n$.

• Ассоциативные мономы $x_i x_j x_s$ длины 3. Таблица n^3 коэффициентов записывается кубом $n \times n \times n$ коэффициентов.

• Неассоциативные мономы $(x_i x_j)x_s$, $x_i(x_j x_s)$ длины 3. Две таблицы n^3 коэффициентов записываются $C_{k-1} = C_2 = 2$ ($(k-1)$-е число Каталана) кубами $n \times n \times n$ коэффициентов.

Единицу и слова свободного группоида также удобно представлять в виде многомерных таблиц. Обозначим $\overline{x} \times \ldots \times \overline{x}$ k-мерную таблицу, в которой на месте $(i_1, \ldots, i_k)$ находится элемент $x_{i_1} \ldots x_{i_k}$. В неассоциативном случае расстановка скобок в произведении $\overline{x} \times \ldots \times \overline{x}$ совпадает с расстановкой скобок в элементе $x_{i_1} \ldots x_{i_k}$. Считая умножение между таблицами коэффициентов и слов, как и сложение между элементами одного из подпространств, описанных выше, поэлементным, получим более удобное представление элементов свободных алгебр.

Примеры:

• элемент длины 2 свободной алгебры:

$$h = a_{x_1} x_1 + \ldots + a_{x_n} x_n + a_{x_1 x_1} x_1 x_1 + a_{x_1 x_2} x_1 x_2 + \ldots + a_{x_n x_n} x_n x_n$$

$$h = \begin{pmatrix} a_{x_1} x_1 \\ \vdots \\ a_{x_n} x_n \end{pmatrix} + \begin{pmatrix} a_{x_1 x_1} x_1 x_1 & \cdots & a_{x_1 x_n} x_1 x_n \\ \vdots & \ddots & \vdots \\ a_{x_n x_1} x_n x_1 & \cdots & a_{x_n x_n} x_n x_n \end{pmatrix}$$

$$h = \begin{pmatrix} a_{x_1} \\ \vdots \\ a_{x_n} \end{pmatrix} \overline{x} + \begin{pmatrix} a_{x_1 x_1} & \cdots & a_{x_1 x_n} \\ \vdots & \ddots & \vdots \\ a_{x_n x_1} & \cdots & a_{x_n x_n} \end{pmatrix} \overline{x} \times \overline{x} =$$

$$= a_{\overline{x}} \cdot \overline{x} + a_{\overline{x} \times \overline{x}} \cdot \overline{x} \times \overline{x}$$

• элемент длины 3 свободной неассоциативной алгебры:

$$h = a_{\overline{x}} \cdot \overline{x} + a_{\overline{x} \times \overline{x}} \cdot \overline{x} \times \overline{x} + a_{\overline{x} \times (\overline{x} \times \overline{x})} \cdot \overline{x} \times (\overline{x} \times \overline{x}) + a_{(\overline{x} \times \overline{x}) \times \overline{x}} \cdot (\overline{x} \times \overline{x}) \times \overline{x}$$

Здесь $a_{((\ldots)\ldots)}$ – все коэффициенты при мономах длины k с указанной скобочной структурой, запи-

санные в виде k-мерной таблицы. Такие таблицы неудобно записывать для $k > 2$, но достаточно легко представлять.

3. ВЕКТОРНЫЕ “СЛОИ”

Покажем еще одно разложение свободной алгебры в прямую сумму на примере следующей алгебры. Рассмотрим алгебру с базой из всех возможных слов заданной длины k с заданным распределением скобок с алфавитом из n символов:

$$A = \left\{ \sum_{(i_1,\ldots,i_k)} \alpha_{i_1\ldots i_k} ((x_{i_1}\ldots x_{i_j})\ldots x_{i_k}) \right\},$$

$\alpha_{i_1\ldots i_k} \in F$, $x_i \in X$, $0 \leqslant i_s \leqslant n$. Зафиксируем один из символов, стоящий на j-м месте, и будем рассматривать все возможные комбинации остальных символов слова. В зависимости от значения зафиксированного символа будут получаться прямые слагаемые:

$$A \cong \left\{ \sum \alpha_{i_1\ldots 1\ldots i_k} ((x_{i_1}\ldots x_1)\ldots x_{i_k}) \right\} \oplus \ldots$$
$$\ldots \oplus \left\{ \sum \alpha_{i_1\ldots n\ldots i_k} ((x_{i_1}\ldots x_n)\ldots x_{i_k}) \right\}.$$

В случае 2-мерных матриц мы оперируем понятиями “строк” и “столбцов” матрицы. Пронумеруем стороны этих таблиц так, что в 2-мерном случае строки индексируются вдоль первой стороны, а столбцы – вдоль второй. В трехмерном случае вдоль первой стороны индексируются горизонтальные “слои”.

Пусть $a_{((\bar{x}\ldots\times\bar{x})\ldots\bar{x})}$ – k-мерная таблица коэффициентов. Обозначим i-й “слой” этой таблицы, индексированный по одной из ее сторон, $a_{((\bar{x}\ldots\times x_i)\ldots\bar{x})}$. Это $(k-1)$-мерная таблица с коэффициентами.

Для примера покажем, как записывается представление $A_{\bar{x}\times(\bar{x}\times\bar{x})} \cdot \bar{x} \times (\bar{x} \times \bar{x})$ при $\bar{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. Размер таблицы равен $2 \times 2 \times 2$. Запишем ее “слои”, индексированные вдоль третьей стороны (“глубины”), в виде расширенных квадратных матриц.

$$A_{\bar{x}\times(\bar{x}\times\bar{x})} = \left(A_{\bar{x}\times(\bar{x}\times x_1)} \mid A_{\bar{x}\times(\bar{x}\times x_2)} \right) =$$
$$= \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \\ a_{x_2(x_1x_1)} & a_{x_2(x_2x_1)} & a_{x_2(x_1x_2)} & a_{x_2(x_2x_2)} \end{array} \right)$$
$$\bar{x} \times (\bar{x} \times \bar{x}) =$$
$$= \left(\begin{array}{cc|cc} x_1(x_1x_1) & x_1(x_2x_1) & x_1(x_1x_2) & x_1(x_2x_2) \\ x_2(x_1x_1) & x_2(x_2x_1) & x_2(x_1x_2) & x_2(x_2x_2) \end{array} \right)$$

Отметим, что часто удобнее рассматривать “слои”, индексированные вдоль первой стороны (“высоты”):

$$A_{x_1\times(\bar{x}\times\bar{x})} = \begin{pmatrix} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} \\ a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \end{pmatrix}$$
$$A_{x_2\times(\bar{x}\times\bar{x})} = \begin{pmatrix} a_{x_2(x_1x_1)} & a_{x_2(x_2x_1)} \\ a_{x_2(x_1x_2)} & a_{x_2(x_2x_2)} \end{pmatrix}$$

4. ТЕХНИКА СВОБОДНОГО ДИФФЕРЕНЦИАЛЬНОГО ИСЧИСЛЕНИЯ В СВОБОДНЫХ НЕАССОЦИАТИВНЫХ АЛГЕБРАХ

При дифференцировании мономов длины 1 получим такой же вектор свободных членов частных производных. Дифференцирование всех мономов длины 2 с коэффициентами, записанными в виде матрицы, даст нам такую же матрицу коэффициентов при правых производных и транспонированную при левых, у которой каждый слой соответствует коэффициентам частных производных. В общем случае при дифференцировании монома длины k получается k различных базисных монома универсальной мультпликативной обертывающей алгебры.

Рассмотрим теперь в алгебре $U(A)$ частные производные, представленные в описанном выше виде:

$$\frac{\partial}{\partial x_i} = a_{x_i} + a_{x_i\times\bar{x}} \cdot r_{\bar{x}} + a_{\bar{x}\times x_i} \cdot l_{\bar{x}} +$$
$$+ a_{x_i\times(\bar{x}\times\bar{x})} \cdot r_{\bar{x}\times\bar{x}} + a_{\bar{x}\times(x_i\times\bar{x})} \cdot r_{\bar{x}} \times l_{\bar{x}} +$$
$$+ a_{\bar{x}\times(\bar{x}\times x_i)} \cdot l_{\bar{x}} \times l_{\bar{x}} + a_{(x_i\times\bar{x})\times\bar{x}} \cdot r_{\bar{x}} \times r_{\bar{x}} +$$
$$+ a_{(\bar{x}\times x_i)\times\bar{x}} \cdot l_{\bar{x}} \times r_{\bar{x}} + a_{(\bar{x}\times\bar{x})\times x_i} \cdot l_{\bar{x}\times\bar{x}} + \ldots$$

Из матриц $a_{((\bar{x}\ldots\times x_i)\ldots\bar{x})}$ можно составить векторы длины n, сгруппировав их по структуре слов алгебры $U(A)$.

Например, для мономов вида $l_{\bar{x}} \times r_{\bar{x}}$ получим вектор слоев коэффициентов:

$$\begin{pmatrix} a_{(\bar{x}\times x_1)\times\bar{x}} \\ \ldots \\ a_{(\bar{x}\times x_n)\times\bar{x}} \end{pmatrix}$$

или, что то же самое, исходную матрицу коэффициентов при $(\bar{x} \times \bar{x}) \times \bar{x}$, с другим порядком сторон.

4.1. Пример дифференцирования группы мономов длины 3 с двумя образующими

Определим “вклад” группы $a_{\bar{x}\times(\bar{x}\times\bar{x})} \cdot \bar{x} \times (\bar{x} \times \bar{x})$ при

$$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

в выражения для частных производных.

$$\mathcal{D}(a_{\bar{x}\times(\bar{x}\times\bar{x})} \cdot \bar{x} \times (\bar{x} \times \bar{x})) =$$

$$= a_{\bar{x}\times(\bar{x}\times\bar{x})} \cdot \begin{pmatrix} \mathcal{D}(x) \times (\bar{x} \times \bar{x}) + \\ + \bar{x} \times (\mathcal{D}(\bar{x}) \times \bar{x}) + \bar{x} \times (\bar{x} \times \mathcal{D}(\bar{x})) \end{pmatrix} =$$

$$= \begin{pmatrix} a_{x_1\times(\bar{x}\times\bar{x})} \\ a_{x_2\times(\bar{x}\times\bar{x})} \end{pmatrix} \cdot \mathcal{D}(x) \times r_{\bar{x}\times\bar{x}} + \begin{pmatrix} a_{\bar{x}\times(x_1\times\bar{x})} \\ a_{\bar{x}\times(x_2\times\bar{x})} \end{pmatrix} \times$$

$$\times \mathcal{D}(x) \times r_{\bar{x}} \times l_{\bar{x}} + \begin{pmatrix} a_{\bar{x}\times(\bar{x}\times x_1)} \\ a_{\bar{x}\times(\bar{x}\times x_2)} \end{pmatrix} \cdot \mathcal{D}(x) \times l_{\bar{x}} \times l_{\bar{x}}$$

$$\begin{pmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \end{pmatrix} = \begin{pmatrix} a_{x_1\times(\bar{x}\times\bar{x})} \\ a_{x_2\times(\bar{x}\times\bar{x})} \end{pmatrix} \cdot r_{\bar{x}\times\bar{x}} + \begin{pmatrix} a_{\bar{x}\times(x_1\times\bar{x})} \\ a_{\bar{x}\times(x_2\times\bar{x})} \end{pmatrix} \times$$

$$\times r_{\bar{x}} \times l_{\bar{x}} + \begin{pmatrix} a_{\bar{x}\times(\bar{x}\times x_1)} \\ a_{\bar{x}\times(\bar{x}\times x_2)} \end{pmatrix} \cdot l_{\bar{x}} \times l_{\bar{x}}$$

$$\begin{pmatrix} a_{x_1\times(\bar{x}\times\bar{x})} \\ a_{x_2\times(\bar{x}\times\bar{x})} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} \\ a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \end{pmatrix} \\ \begin{pmatrix} a_{x_2(x_1x_1)} & a_{x_2(x_2x_1)} \\ a_{x_2(x_1x_2)} & a_{x_2(x_2x_2)} \end{pmatrix} \end{pmatrix} \sim$$

$$\sim \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \\ a_{x_2(x_1x_1)} & a_{x_2(x_2x_1)} & a_{x_2(x_1x_2)} & a_{x_2(x_2x_2)} \end{array}\right)$$

$$\begin{pmatrix} a_{\bar{x}\times(x_1\times\bar{x})} \\ a_{\bar{x}\times(x_2\times\bar{x})} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} \\ a_{x_1(x_1x_2)} & a_{x_2(x_1x_2)} \end{pmatrix} \\ \begin{pmatrix} a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} \\ a_{x_1(x_2x_2)} & a_{x_2(x_2x_2)} \end{pmatrix} \end{pmatrix} \sim$$

$$\sim \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} & a_{x_1(x_1x_2)} & a_{x_2(x_1x_2)} \\ a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} & a_{x_1(x_2x_2)} & a_{x_2(x_2x_2)} \end{array}\right)$$

$$\begin{pmatrix} a_{\bar{x}\times(\bar{x}\times x_1)} \\ a_{\bar{x}\times(\bar{x}\times x_2)} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} \\ a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} \end{pmatrix} \\ \begin{pmatrix} a_{x_1(x_1x_2)} & a_{x_2(x_1x_2)} \\ a_{x_1(x_2x_2)} & a_{x_2(x_2x_2)} \end{pmatrix} \end{pmatrix} \sim$$

$$\sim \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} \\ a_{x_1(x_1x_2)} & a_{x_2(x_1x_2)} & a_{x_1(x_2x_2)} & a_{x_2(x_2x_2)} \end{array}\right)$$

Так, дифференцирование 8 неассоциативных мономов с заданной скобочной структурой дает нам 24 ассоциативных монома алгебры $U(A)$ в выражениях частных производных

5. КРИТЕРИЙ ПРИМИТИВНОСТИ ЭЛЕМЕНТА ДЛИНЫ 2 И 3 СВОБОДНОЙ НЕАССОЦИАТИВНОЙ АЛГЕБРЫ С ДВУМЯ ОБРАЗУЮЩИМИ

Приведенные выше рассуждения позволяют нам сформулировать следующий критерий примитивности элементов длины 2 в терминах линейной алгебры.

Предложение 1. *Элемент* $h = a \cdot \bar{x} + B \cdot \bar{x} \times \bar{x} + C \cdot (\bar{x} \times \bar{x}) \times \bar{x} + D \cdot \bar{x} \times (\bar{x} \times \bar{x})$, *примитивен тогда и только тогда, когда ранг матрицы* $(a|B|B^T|C|C^T|D|D^T)$ *больше ранга матрицы* $(B \mid B^T \mid C \mid C^T \mid D \mid D^T)$.

Доказательство. Воспользуемся техникой свободного дифференциального исчисления и критерием примитивности.

Элемент примитивен тогда и только тогда, когда найдутся такие $m_1, m_2, \ldots, m_n \in U(A)$, что

$$m_1 \frac{\partial}{\partial x_1} + m_2 \frac{\partial}{\partial x_2} + \ldots + m_n \frac{\partial}{\partial x_n} = 1.$$

Матрица $(a|B|B^T|\ldots)$ содержит n строк, где n – число свободных образующих. Каждая из строк матрицы соответствует представлению частной производной по одной из переменных в виде многомерных таблиц.

Задача определения примитивности сводится к алгоритму редукции, шагом которого является устранение старших мономов из производных за счет других производных. Поскольку все старшие мономы l_{x_i} и r_{x_i} производных элементов длины 2 (в общем виде будем записывать op_{x_i}) могут быть получены, либо как $a \cdot op_{x_i} = b \cdot (c \cdot op_{x_i})$, либо как $a \cdot op_{x_i} = (b \cdot op_{x_i}) \cdot c$, возможна только линейная редукция (то есть с коэффициентом из F). Аналогично, среди мономов производной элемента длины 3 обязательно встретится $op_{x_i x_j}$ для которого возможна только линейная редукция.

Итак, для редукции используется только умножение на элементы F. В этом случае алгоритм редукции сводится к решению системы линейных уравнений над полем F.

Воспользовавшись критерием Кронекера–Капелли, получаем доказательство утверждения.

Полученное в ходе доказательства утверждение можно сформулировать в виде леммы.

Лемма 1. *Если среди членов неассоциативного элемента длины k встречаются мономы вида* $x_i \cdot w$ *или* $w \cdot x_i$*, то среди соответствующих им мономов частных производных встречаются такие, для которых возможна только линейная редукция.*

Элемент с такими членами примитивен тогда и только тогда, когда выполняются условия, аналогичные (с учетом большего числа матриц коэффициентов) условиям доказанного критерия примитивности.

Доказательство. К доказанному выше осталось доказать только необходимость выполнения условий. После линейной редукции возможны два случая:

1) одно из выражений частных производных стало константой (то есть элемент примитивен);

2) одно из выражений стало меньшей длины, но не стало константой, другое всё ещё содержит моном вида op_w, из чего следует, что дальнейшая редукция невозможна.

6. ПРИЗНАК ПРИМИТИВНОСТИ

Полученный для элементов длины 2 и 3 критерий можно обобщить до признака примитивности элементов произвольной длины. Транспонирование матриц при этом заменяется на перестановку сторон таблиц коэффициентов. А вместо ранга матрицы необходимо рассматривать ранг системы векторов, составленных из элементов слоев многомерных таблиц коэффициентов.

Под действие этого признака попадает только случай, когда производится линейная редукция системы частных производных. То есть решается система линейных уравнений:

$$\alpha_1 \frac{\partial}{\partial x_1} + \ldots + \alpha_n \frac{\partial}{\partial x_n} = 1,\ \alpha_1,\ldots,\alpha_n \in F.$$

7. ОЦЕНКА ЧИСЛА ПРИМИТИВНЫХ ЭЛЕМЕНТОВ С ДВУМЯ ОБРАЗУЮЩИМИ ПРОИЗВОЛЬНОЙ ДЛИНЫ

7.1. Частные производные одной группы мономов

Проиллюстрируем алгоритм расчета числа примитивных элементов на примере рассмотренной выше группы мономов $a_{\overline{x}\times(\overline{x}\times\overline{x})} \cdot \overline{x} \times (\overline{x} \times \overline{x})$. Пусть $\frac{\partial}{\partial x_2} = \alpha + t\frac{\partial}{\partial x_1}$, $\alpha, t \in F$. Тогда

$$\left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \\ a_{x_2(x_1x_1)} & a_{x_2(x_2x_1)} & a_{x_2(x_1x_2)} & a_{x_2(x_2x_2)} \end{array}\right) =$$

$$= \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_1(x_1x_2)} & a_{x_1(x_2x_2)} \\ ta_{x_1(x_1x_1)} & ta_{x_1(x_2x_1)} & ta_{x_1(x_1x_2)} & ta_{x_1(x_2x_2)} \end{array}\right)$$

$$\left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} \\ a_{x_1(x_1x_2)} & a_{x_2(x_1x_2)} & a_{x_1(x_2x_2)} & a_{x_2(x_2x_2)} \end{array}\right) =$$

$$= \left(\begin{array}{cc|cc} a_{x_1(x_1x_1)} & a_{x_2(x_1x_1)} & a_{x_1(x_2x_1)} & a_{x_2(x_2x_1)} \\ ta_{x_1(x_1x_1)} & ta_{x_2(x_1x_1)} & ta_{x_1(x_2x_1)} & ta_{x_2(x_2x_1)} \end{array}\right)$$

$$a_{x_2(x_2x_2)} = ta_{x_2(x_2x_1)} = t^2 a_{x_1(x_2x_1)} =$$

$$= ta_{x_2(x_1x_2)} = t^2 a_{x_2(x_1x_1)} =$$

$$= ta_{x_1(x_2x_2)} = t^2 a_{x_1(x_1x_2)} = t^3 a_{x_1(x_1x_1)},$$

где t – общий коэффициент пропорциональности между неединичными мономами частных производных. Это равносильно тому, что t – коэффициент пропорциональности при мономах неассоциативной алгебры с одинаковой расстановкой скобок. При выражении $a_{x_2(x_2x_2)}$ через другие коэффициенты его степень равна количеству x_1 в мономе, соответствующем коэффициенту, и обратно, если выражать

$$a_{x_1(x_1x_1)} = \ldots = s^3 a_{x_2(x_2x_2)}.$$

7.2. Число примитивных элементов

Полученное правило справедливо для мономов любой длины свободной неассоциативной алгебры над конечным полем. Если матрица коэффициентов при какой-либо группе мономов длины m записывается m-мерной таблицей $(a_{i_1 i_2 \ldots i_m})$, $i_j \in \{1,2\}$, то либо $a_{11\ldots1} = \ldots = t^m a_{22\ldots2}$, либо $a_{22\ldots2} = \ldots = s^m a_{11\ldots1}$. Это позволяет нам для каждой группы мономов задать лишь один из коэффициентов. Остальные получаются умножением на выбранный коэффициент t.

Обозначим через $S_2^k(q)$ число примитивных элементов длины k свободной неассоциативной алгебры с двумя образующими над конечным полем F_q.

Линейная часть примитивного элемента может быть получена $q^2 - 1 = q(q-1)$ способами.

Для каждого варианта линейной части оценим число способов получения элемента длины k, удовлетворяющего полученным соотношениям. Коэффициент пропорциональности t можно выбрать q способами, при этом в случае $t = 0$ у нас еще возникает необходимость для каждой группы мономов

длины m выбрать $a_{x_2(\dots x_2)} = \dots = t^m a_{x_1(\dots x_1)}$ или $a_{x_1(\dots x_1)} = \dots = t^m a_{x_2(\dots x_2)}$. Включив этот выбор в число способов выбора t, получим $q+1$ вариант.

Далее оценим число способов выбрать коэффициенты при мономах $x_2(\dots x_2)$ либо $x_1(\dots x_1)$ (в зависимости от выбора t) всех групп длины m. Число вариантов расстановки скобок в мономе длины m равно $(m-1)$-му числу Каталана C_{m-1}, число вариантов выбрать в каждой группе коэффициент равно q. В итоге получаем $q^{C_{m-1}}$ вариант. Чтобы длина элемента была равна k, необходимо, чтобы хотя бы один коэффициент при мономах длины k был ненулевым, что даёт нам $q^{C_{k-1}} - 1$ вариант выбора коэффициента при мономах длины k. Суммарное число комбинаций коэффициентов для мономов длин $2 \dots k-1$ равно $q^{C_1} \cdot \dots \cdot q^{C_{k-2}} = q^{\sum_{m=2}^{k-1} C_{m-1}}$.

Итак, мы готовы записать оценку для $S_2^k(q)$.

$$S_2^k(q) \geqslant \underbrace{(q+1)}_{t} \cdot \underbrace{q(q-1)}_{m=1} \cdot \underbrace{q^{\sum_{m=2}^{k-1} C_{m-1}}}_{m=2\dots k-1} \cdot \underbrace{(q^{C_{k-1}} - 1)}_{m=k}$$

Считая, что $C_0 = 1$, можно сократить запись:

$$S_2^k(q) \geqslant (q+1)(q-1) q^{\sum_{m=1}^{k-1} C_{m-1}} (q^{C_{k-1}} - 1).$$

7.3. Формулы для некоторых длин

Используя, полученную формулу запишем оценки для некоторых длин:

$$S_2^2(q) = (q+1)(q-1)q^1(q^1 - 1) = q(q-1)^2(q+1)$$

$$S_2^3(q) = (q+1)(q-1)q^2(q^2 - 1) = q^2(q-1)^2(q+1)^2$$

$$S_2^4(q) \geqslant (q+1)(q-1)q^4(q^5 - 1)$$

$$S_2^5(q) \geqslant (q+1)(q-1)q^9(q^{14} - 1)$$

Для $S_2^2(q)$ и $S_2^3(q)$ равенство обусловлено тем, что вместо признака можно использовать критерий примитивности.

Правая часть полученной оценки в случае длин 2 и 3 совпадает с формулами подсчета таких элементов, полученными ранее А. А. Чеповским в диссертации [5].

8. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМОВ

Ранее для исследования примитивных элементов были реализованы алгоритмы[3] работы с примитивными элементами в свободных неассоциативных алгебрах в системе компьютерной алгебры SageMath. Для небольших конечных полей с помощью теста примитивности были посчитаны примитивные элементы небольших длин. Полученные выше формулы дают те же значения для тех же полей и длин.

Описанное представление позволяет оптимизировать алгоритмы работы с примитивными элементами, заменив ресурсоемкие операции на работу с таблицами: вместо символьного дифференцирования производить транспонирование матриц, упростить умножение элементов алгебр.

8.1. Подсчет количества примитивных элементов длины 4 в свободной неассоциативной алгебре с двумя образующими

Выше мы показали, что если элемент содержит один из мономов вида $x(x(xx))$, $x((xx)x)$, $(x(xx))x$, $((xx)x)x$, то в алгоритме проверки примитивности элемента будут использованы только линейные редукции. Все такие примитивные элементы можно посчитать по полученной выше формуле. Значит, для получения общего числа примитивных элементов длины 4 необходимо посчитать те, которые требуют умножения на нескалярные элементы алгебры $U(A)$ в ходе редукции. Все такие элементы будут содержать мономы вида $(xx)(xx)$.

Найдем частные производные мономов такого вида:

$$\mathcal{D}((\overline{xx})(\overline{xx})) = r_{\overline{xx}} r_{\bar{x}} \mathcal{D}(\bar{x}) + \\ + r_{\overline{xx}} l_{\bar{x}} \mathcal{D}(\bar{x}) + l_{\overline{xx}} r_{\bar{x}} \mathcal{D}(\bar{x}) + l_{\overline{xx}} l_{\bar{x}} \mathcal{D}(\bar{x}).$$

Пронумеруем получившиеся мономы алгебры $U(A)$ по порядку переменной в мономе $(\overline{xx})(\overline{xx})$, по которой при дифференцировании получается $\mathcal{D}(\bar{x})$. Получим $1 : r_{\overline{xx}} r_{\bar{x}}$, $2 : r_{\overline{xx}} l_{\bar{x}}$, $3 : l_{\overline{xx}} r_{\bar{x}}$, $4 : l_{\overline{xx}} l_{\bar{x}}$.

Если таблицу мономов вида $(\overline{xx})(\overline{xx})$ записать в виде

$$\left(\begin{array}{cc|cc} (xx)(xx) & (xx)(xy) & (xy)(xx) & (xy)(xy) \\ (xx)(yx) & (xx)(yy) & (xy)(yx) & (xy)(yy) \\ \hline (yx)(xx) & (yx)(xy) & (yy)(xx) & (yy)(xy) \\ (yx)(yx) & (yx)(yy) & (yy)(yx) & (yy)(yy) \end{array}\right),$$

то, используя введённые обозначения, частные производные $\partial/(\partial x)$ и $\partial/(\partial y)$ можно записать соответственно таблицами:

$$\left(\begin{array}{ccc|c} 1234 & 123 & 134 & 13 \\ 124 & 12 & 14 & 1 \\ \hline 234 & 23 & 34 & 3 \\ 24 & 2 & 4 & \end{array}\right) \text{ и } \left(\begin{array}{ccc|c} & & 4 & 2 \quad 24 \\ & 3 & 34 & 23 \quad 234 \\ \hline 1 & 14 & 12 & 124 \\ 13 & 134 & 123 & 1234 \end{array}\right).$$

Считая, что

• D – таблица коэффициентов, соответствующих мономам в этих таблицах мономов;

• при использовании ее в качестве коэффициентов при частных производных первый индекс указывает на то, к какой производной относится коэффициент (например, d_{1ijk} – коэффициент при некотором мономе $\partial/(\partial x)$, а d_{2ijk} – коэффициент при некотором мономе $\partial/(\partial y)$);

• D^σ – транспонированная таблица коэффициентов, с помощью перестановки σ, то есть $d^\sigma_{i_1i_2i_3i_4} = d_{i_{\sigma(1)}i_{\sigma(2)}i_{\sigma(3)}i_{\sigma(4)}}$;

• из монома $(x_1x_2)(x_3x_4)$ алгебры A получаются следующие мономы алгебры $U(A)$: $r_{x_3x_4}r_{x_2}$, $r_{x_3x_4}l_{x_1}$, $l_{x_1x_2}r_{x_4}$, $l_{x_1x_2}l_{x_3}$, и соответствующие им перестановки (234), (12334), (321), (4321),

получим следующие выражения частных производных группы мономов $(\overline{xx})(\overline{xx})$: $D^{(234)}r_{\overline{xx}}r_{\bar{x}} + D^{(1234)}r_{\overline{xx}}l_{\bar{x}} + D^{(321)}l_{\overline{xx}}r_{\bar{x}} + D^{(4321)}l_{\overline{xx}}l_{\bar{x}}$.

Если примитивный элемент содержит моном вида $(\overline{xx})(\overline{xx})$, частные производные которого требуют редукции, то он не может содержать мономы длины 3. Так как в этом случае в выражения частных производных будут входить мономы вида r_{xx} или l_{xx}, которые могут быть редуцированы только такими же мономами. Итак, примитивный элемент длины 4, не подходящий под условия признака примитивности, имеет вид: $A\overline{x} + B\overline{xx} + D(\overline{xx})(\overline{xx})$.

Также справедливы соотношения для старших мономов, полученные ранее:

$$s^4d_{(x_2x_2)(x_2x_2)} = s^3td_{(x_2x_2)(x_2x_1)} =$$
$$= s^3td_{(x_2x_2)(x_1x_2)} = s^3td_{(x_2x_1)(x_2x_2)} =$$
$$= s^3td_{(x_1x_2)(x_2x_2)} = s^2t^2d_{(x_2x_2)(x_1x_1)} =$$
$$= s^2t^2d_{(x_2x_1)(x_2x_1)} = s^2t^2d_{(x_1x_2)(x_2x_1)} =$$
$$= s^2t^2d_{(x_2x_1)(x_1x_2)} = s^2t^2d_{(x_1x_2)(x_1x_2)} =$$
$$= s^2t^2d_{(x_1x_1)(x_2x_2)} = st^3d_{(x_2x_1)(x_1x_1)} =$$
$$= st^3d_{(x_1x_2)(x_1x_1)} = st^3d_{(x_1x_1)(x_2x_1)} =$$
$$= st^3d_{(x_1x_1)(x_1x_2)} = t^4d_{(x_1x_1)(x_1x_1)}.$$

Исходя из этого получим, что подходящих наборов коэффициентов при таких мономах в каждом поле будет $(q+1)(q-1)$, как уже было показано выше.

Посчитаем для некоторых конечных полей для каждого такого набора число примитивных элементов методами компьютерной алгебры.

С учетом всех приведенных выше рассуждений для подсчета элементов длины 4 в свободных неассоциативных алгебрах с двумя образующими над конечными полями получим следующие алгоритмы.

Алгоритм 1. *Генерация таблиц коэффициентов для мономов длины 4.*

1) подготовить наборы индексов:
   1 – 0000;
   2 – 1000, 0100, 0010, 0001;
   4 – 0111, 1011, 1101, 1110;
   5 – 1111;
   3 – остальные 6 комбинаций;
2) установить значение множителя $t := 0$;
3) если $t > q$, перейти к шагу 12;
4) подготовить таблицы-шаблоны с множителями, поместив t^{k-1} по индексам набора k; если $t = q$, то наоборот, по индексам набора k поместить значение t^{6-k};
5) установить опорное значение $a := 1$;
6) если $a \geqslant q$, перейти к шагу 10;
7) сформировать таблицу коэффициентов как таблицу-шаблон, умноженную на a;
8) увеличить $a := a + 1$;
9) перейти к шагу 6;
10) увеличить $t := t + 1$;
11) перейти к шагу 3;
12) вернуть все сформированные таблицы коэффициентов;
13) завершить алгоритм.

Алгоритм 2. *Генерация таблиц коэффициентов для мономов длин 1 и 2.*

1) подготовить список из 6 нулевых элементов;
2) установить $i := 1$;
3) если $i > 6$ перейти к шагу 11;
4) увеличить элемент i в списке на 1;
5) если значение элемента i меньше q, то перейти к шагу 9;
6) установить элемент i в списке равным 0;
7) увеличить $i := i + 1$;
8) перейти к шагу 3;
9) если в списке есть ненулевые элементы, то сформировать вектор коэффициентов при мономах длины 1 из первых двух элементов списка, матрицу коэффициентов при мономах длины 2 из последних четырех элементов списка;

10) перейти к шагу 2;

11) вернуть все сформированные таблицы коэффициентов;

12) завершить алгоритм.

Алгоритм 3. *Проверка примитивности элементов для заданной таблицы коэффициентов при мономах длины 4.*

Вход: два выражения p_1 и p_2. На каждом этапе редукция описывается следующими шагами:

1) найти коэффициенты k_1 и k_2, не равные одновременно нулю в соответствующих значениях частных производных для старшего монома;

2) в качестве редуцируемого выражения p_1 выбрать то, в котором найден ненулевой коэффициент;

3) вычислить выражение $p_2 := k_1 p_2 + k_2 p_1 \bmod q$;

4) если в выражении p_2 сохранились старшие мономы той же степени, значит редукция невозможна, перейти к шагу 10;

5) найти коэффициенты в соответствующих друг другу группах мономов: k_2 – ненулевые в старших мономах p_2 и k_1 – ненулевые в старших мономах p_1;

6) найти k_2 как $k_1 \times k_2$, в качестве нового значения k_2 выбрать любой ненулевой элемент k_2;

7) произвести редукцию $p_1 := p_2$, $p_2 := k_1 p_2 + k_2 p_1 \bmod q$;

8) если в выражении p_2 сохранились старшие мономы той же степени, значит редукция невозможна, перейти к шагу 10;

9) вернуть p_1, p_2 и завершить алгоритм с кодом УСПЕХ;

10) вернуть исходные выражения, завершить алгоритм с кодом НЕУСПЕХ.

Описанное на шаге 6 алгоритма 3 произведение определим следующим образом. Пусть a – таблица размера $t_1 \times t_2 \times \ldots \times t_\alpha$, b – таблица размера $s_1 \times s_2 \times \ldots \times s_\beta$. Тогда результатом произведения будет таблица $c = a \times b$ размера $t_1 \times t_2 \times \ldots \times t_\alpha \times s_1 \times s_2 \times \ldots \times s_\beta$ такая, что

$$c_{i_1 \ldots i_\alpha j_1 \ldots j_\beta} = a_{i_1 \ldots i_\alpha} \cdot b_{j_1 \ldots j_\beta}.$$

Эта и другие операции с многомерными таблицами реализованы в пакетах NumPy, PyTorch, TensorFlow языка Python. При этом PyTorch и TensorFlow поддерживают вычисления на GPU, что позволяет многократно ускорить обработку больших объёмов данных.

Алгоритм 4. *Подсчет примитивных элементов длины 4.*

1) перебрать простые числа: 2, 3, 5, 7, 11, ...;

2) с помощью **Алгоритма 1** сформировать таблицы коэффициентов при старших мономах по текущему выбранному простому числу;

3) установить счетчик примитивных элементов равным 0;

4) для каждой из таблиц с помощью **Алгоритма 2** сформировать таблицы коэффициентов при остальных мономах;

5) произвести с помощью **Алгоритма 3** редукцию каждого из элементов пока алгоритм возвращает УСПЕХ;

6) если в какой-то момент одно из возвращаемых **Алгоритмом 3** выражений будет ненулевой константой, то элемент примитивен, увеличить счетчик примитивных элементов на 1;

7) вывести по каждому набору коэффициентов старших мономов полученное число примитивных элементов.

В результате работы алгоритма 4 получено, что для каждого набора коэффициентов (вне зависимости от выбранного множителя t и равенства его нулю), число примитивных элементов в простых полях соответственно: $F_2 : 3$, $F_3 : 8$, $F_5 : 24$, $F_7 : 48$, $F_{11} : 120$, ...

Метод неопределённых коэффициентов даёт выражение для этого числа $q^2 - 1$. Тогда общее число примитивных элементов с нередуцируемыми линейно частными производными получится $(q-1)^2(q+1)^2$.

Таким образом, мы получили недостающую часть числа S_2^4:

$$S_2^4 = (q+1)(q-1)q^4(q^5-1) + (q+1)^2(q-1)^2.$$

ИСТОЧНИК ФИНАНСИРОВАНИЯ

При финансовой поддержке Российского научного фонда, грант 22-11-00052.

СПИСОК ЛИТЕРАТУРЫ

1. *Artamonov V.A., Klimakov A.V., Mikhalev A.A., Mikhalev A.V.* Primitive and almost-primitive elements of free algebras of Schreier varieties, Fundam // Prikl. Mat. 2016. V. 21. № 2. P. 3–35.
2. *Kurosh A.G.* Free non-associative algebras and free products of algebras // Mat. Sb. 1947. V. 20. P. 239–262.
3. *Maisuradze M.V.* Software implementation of algorithms for working with primitive elements in free nonassocia-

tive algebras // Intellekt. Sist. Teor. Prilozh. 2021. V. 25. № 4. P. 170–175.
4. *Mikhalev A.A., Mikhalev A.V., Chepovskii A.A., Shampan'er K.* Primitive elements of free associative algebras // Fundam. Prikl. Mat. 2007. V. 13. № 5. P. 171–192.
5. *Chepovskii A.A.* Primitive elements of algebras of Schreier varieties, Cand. Sci. (Phys.-Math.) Dissertation, Moscow: Mosk. Gos. Univ., 2011.
6. *Chepovskii A.A.* Number of primitive elements of lengths 1 and 2 in free non-associative algebras over a finite field // Vestn. Novosib. Gos. Univ. Ser.: Mat., Mekh., Inf. 2011. V. 11. P. 119–122.
7. *Mikhalev A.A., Umirbaev U.U., Yu J.-T.* Automorphic orbits of elements of free non-associative algebras, J. Algebra. 2001. P. 198–223.
8. *Mikhalev A.A., Shpilrain V., Yu J.-T.* Combinatorial Methods: Free Groups, Polynomials, and Free Algebras, New York: Springer, 2004.

PRIMITIVE ELEMENTS OF FREE NON-ASSOCIATIVE ALGEBRAS OVER FINITE FIELDS

© 2024 M. V. Maisuradze[a], A. A. Mikhalev[a]

[a] *Department of Mechanics and Mathematics, Moscow State University, Moscow, 119991 Russia*

The representation of elements of free non-associative algebras as a set of multidimensional tables of coefficients is defined. An operation for finding partial derivatives for elements of free non-associative algebras in the same form is considered. Using this representation, a criterion of primitivity for elements of lengths 2 and 3 in terms of matrix ranks, as well as a primitivity test for elements of arbitrary length, is derived. This test makes it possible to estimate the number of primitive elements in free non-associative algebras with two generators over a finite field. The proposed representation allows us to optimize algorithms for symbolic computations with primitive elements. Using these algorithms, we find the number of primitive elements of length 4 in a free non-associative algebra of rank 2 over a finite field.

Keywords: Schreier variety of linear algebras, free non-associative algebras, primitive elements of free algebras, and free differential calculus in free algebras

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 93–99*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 514.8

ПОРТ-ГАМИЛЬТОНОВЫ СИСТЕМЫ: РАСПОЗНАВАНИЕ СТРУКТУРЫ И ПРИЛОЖЕНИЯ

© 2024 г. В. Н. Сальников[a], *

[a]*ЦНРС и Университет города Ла-Рошель, 17042 Ла-Рошель, Проспект Мишеля Крепо, Франция*

**E-mail: vladimir.salnikov@univ-lr.fr*

Поступила в редакцию 11.08.2023
После доработки 10.08.2023
Принята к публикации 01.10.2023

В данной работе мы продолжаем рассматривать задачу восстановления порт-Гамильтоновой структуры для произвольной системы дифференциальных уравнений. Мы дополняем предыдущую работу по этой теме, объясняя выбор и подробности применения алгоритмов машинного обучения. Мы также объясняем, какие возможности открывает такой подход для потенциально нового определения канонических форм и классификации систем дифференциальных уравнений.

Ключевые слова: геометризация механики, порт-Гамильтоновы системы, методы машинного обучения для ОДУ

DOI: 10.31857/S0132347424020121 **EDN:** RNXHNU

1. ВВЕДЕНИЕ / МОТИВАЦИЯ

Данная статья – продолжение серии работ в рамках глобального проекта по “геометризации механики”. Под этим термином, не обязательно общепринятым, мы понимаем различные подходы, основанные изначально на результатах и конструкциях из дифференциальной или обобщённой геометрии, которые применяются к качественному анализу уравнений, описывающих механические системы. В этот проект входит разработка геометрического формализма, релевантного для максимально широкого класса задач механики. Он также посвящен поиску или построению разумных стратегий моделирования механических систем и численного решения полученных уравнений. Основная прикладная цель проекта – предложить наиболее полный “набор инструментов”, оптимальный в “бытовом” смысле, то есть обеспечивающий достаточно точное решение таких задач с разумными затратами вычислительных ресурсов. Мы приводим некоторый обзор таких методов в [1]: в нем мы показываем, что конструкции из обобщенной и градуированной геометрии играют важную роль.

В [2, 3] мы сформулировали ряд открытых вопросов и задач, потенциально решаемых современными методами компьютерной алгебры; построение порт-Гамильтоновой структуры заданных уравнений и изучение порт-Гамильтовых систем (ПГС – набор Гамильтоновых систем, соединённых по определённым правилам) было одной из них. В [4] мы подробно рассмотрели Гамильтонову составляющую ПГС, которая была существенно связана с результатами из симплектической и Пуассоновой геометрии. Про порты мы лишь упомянули, что их можно удобно описывать с помощью графов, к которым и применяются алгоритмы анализа. В данной статье мы остановимся на этом вопросе подробнее, а именно объясним релевантность подхода к анализу с помощью машинного обучения и приведем некоторые аргументы в пользу выбранных алгоритмов. Мы также остановимся чуть подробнее на некоторой новой философии анализа внутренней структуры произвольных систем дифференциальных уравнений, естественным образом вытекающей из формализма ПГС.

В секции 2 мы напомним определения и обозначения из ПГС, сформулируем их важные свойства. В секции 3 мы опишем “опыт” применения различных подходов к анализу ПГС. Мы закончим статью упоминанием прямых приложений данного подхода, связанных с геометрическими интеграторами в моделировании сложных систем. Мы также обсудим потенциальные приложения для нового подхода к определению канонической формы систем дифференциальных уравнений.

2. ПОРТ-ГАМИЛЬТОНОВЫ СИСТЕМЫ

Идея порт-Гамильтоновых систем достаточно проста и изящна: для нескольких консервативных механических систем, описываемых в рамках классической Гамильтоновой динамики, рассмотрим взаимодействие, заданное некоторыми силами – эти

силы назовём портами. Насколько нам известно, впервые такой подход был подробно описан в [5], с инженерной точки зрения, а затем пересмотрен в [6] с некоторым описанием классификации физики портов. Достаточно общепринятая терминология – называть каждую из таких систем вместе с её входящими и исходящими портами *узлом*.

Напомним, как эта конструкция формализуется математически. Каждый узел можно записать следующим образом:

$$\dot{\mathbf{x}} = (J(\mathbf{x}) - R(\mathbf{x}))\frac{\partial H}{\partial \mathbf{x}} + w(\mathbf{x})\mathbf{u}. \qquad (2.1)$$

Здесь $\mathbf{x}$ – векторная переменная, обозначающая состояние узла, $J(\mathbf{x})$ – кососимметричная матрица, описывающая симплектическую или Пуассонову структуру (см. [4]), $H(\mathbf{x})$ – Гамильтониан; $R(\mathbf{x}),w(\mathbf{x})$ и $\mathbf{u}$ соответствуют внутренним или внешним портам.

Порт-Гамильтонова система составляется из таких узлов соединением через порты: внешние переменные одних узлов зависят от внутренних для других. Для дальнейшего важно понимать ключевое свойство ПГС: несколько узлов, соединённых вместе, снова являются узлом.

Понятно, что “сборка” порт-Гамильтоновой системы из модулей – это формально простая техническая конструкция. Здесь мы обсуждаем задачу, обратную к ней: зная, что система дифференциальных уравнений

$$\dot{\mathbf{X}} = \mathbf{F}(\mathbf{X}) \qquad (2.2)$$

получена как порт-Гамильтонова, восстановить её структуру.

Напомним, что алгоритм, который мы начали объяснять в [4], выглядел (не вдаваясь в подробности) следующим образом.

Алгоритм

Входные данные: система дифференциальных уравнений вида (0.2).

1. Восстановление графа связности ПГС, то есть идентификация узлов и связей между ними.

2. Разметка полученного графа, то есть идентификация или выбор симплектической/Пуассоновой структуры и соответствующего Гамильтониана.

3. Разметка связей между узлами, то есть идентификация портов.

Выходные данные: граф (множество узлов и портов ПГС), размеченный в обозначениях (0.1).

Пункты 2 и 3 были подробно изучены в [4] – после соответствующего анализа они сводятся к символьным вычислениям с дифференциальными формами, векторными и бивекторными полями. В предыдущей статье мы рассматривали примеры с помощью простейших инструментов на языке Python, упомянем здесь лишь, что пакет SageManifolds в SageMath ([7]) оказался удобным и более продвинутым для таких целей.

Что же касается п. 1, в предварительных вычислениях для “proof of concept” мы использовали совершенно прямолинейную реализацию алгоритмов, о которой скажем ниже, но уже тогда было понятно, что вопрос более интересный и заслуживает отдельного рассмотрения. Сформулируем некоторые свойства ПГС, повлиявшие на выбор алгоритма решения этой задачи.

Как уже было упомянуто, важное свойство ПГС – возможность их соединения:

- два соединённых или даже независимых узла могут рассматриваться как один;
- одна переменная может быть узлом;
- вся ПГС может считаться узлом;
- несколько ПГС, соединённых вместе – снова ПГС.

Это значит, что без дополнительных условий задача восстановления структуры ПГС по уравнению вида (2.2) имеет существенно неединственное решение.

Рассмотрим теперь всю ПГС как один узел, иными словами, конфигурацию, когда уравнение (2.2) имеет вид (2.1). В таком случае правые части системы имееют довольно специальный вид. Матрицы J и R отвечают за внутренние степени свободы узла – когда узел сообран из нескольких, они будут иметь блочную структуру: подматрицы меньшего размера будут сворачиваться с градиентом Гамильтониана, зависящего от некоторого подмножества переменных. Переменные w и $\mathbf{u}$, наоборот, будут “перемешивать” блоки и встречаться парами: внутренние для одного – внешние для другого подузла.

3. ПОДХОДЫ К ОПРЕДЕЛЕНИЮ СТРУКТУРЫ ГРАФА

В этой секции мы представим три различных подхода, которые мы рассматривали для определения структуры графа ПГС. Мы опишем в том числе и “неудачный” опыт, а именно стратегии, которые не привели к желаемому или оптимальному результату. В свете замечаний выше, мы объясним различия методов и причины возникающих сложностей. Мы постараемся сделать это с наименьшим количеством технических подробностей. Напомним, цель в сис-

теме уравнений вида (2.2) сгруппировать переменные в узлы и восстановить связи между ними.

Детерминистский подход

Описанная структуры матриц J и R кажется довольно "узнаваемой" – естественно возникает идея восстановить эту структуру *точно*, то есть сформулировать набор критериев, позволяющих классифицировать каждое слагаемое уравнения (2.2). Идея алгоритма могла бы быть следующая: в каждом уравнении системы (2.2) каждое из слагаемых отнести либо к переменным "своего" узла, либо к портам по некоторому признаку, затем проверить совместность результата с учётом Гамильтоновой структуры узлов.

На деле единственное решение, которое получилось формализовать – перебрать *все возможные комбинации* размеров узлов и заполнить их *всеми возможными способами* слагаемыми из (2.2). При этом шаг проверки совместности сводится к описанному в [4] восстановлению Пуассоновой или симплектической структуры (матрицы J) и подбору Гамильтониана, то есть аналитическому решению нелинейной системы дифференциальных уравнений. Только в отличие от оригинального подхода [4], разделение на порты и внутренние переменные в каждом случае будет зафиксировано, что систематически приводит именно к несовместным конфигурациям. Это же ограничение приводит к невозможности корректировать Гамильтониан, если он оказывается слишком сложным и/или сильно нелинейным для восстановления. Все эти явления были заметны на очень простых системах в малой размерности: достаточно рассмотреть гармонические осцилляторы с минимально нелинейным взаимодействием, порт-Гамильтонова структура для которых легко читается невооруженным глазом. Очевидно также, что для больших систем даже перебор размеров блоков становится неразумным. Таким образом даже до попыток настоящей программной реализации стало понятна ограниченность такого подхода.

Причина трудностей каждый раз одна и та же: неоднозначность представления системы вида (2.2) в виде ПГС. В [4] мы объяснили, что именно эта неоднозначность должна облегчать процесс, предоставляя свободу выбора для идентификации геометрических структур. Это же свойство оказалось удобно использовать и для построения узлов в двух методах, описанных далее.

Машинное обучение

Как мы уже не раз упомянули, по-настоящему работающие и удобные методы оказались связаны с довольно стандартными подходами машинного обучения с использованием нейронных сетей. Мы, естественно, не собираемся излагать здесь хоть сколько-то подробно идеи нейронных сетей: даже обзор литературы занял бы неразумно большое пространство. Ограничимся лишь напоминанием ставшего общепринятым подхода. Машинное обучение применяется в задачах принятия решений в случае, когда есть возможность проанализировать множество похожих задач в предварительной фазе "обучения" сети. При этом ответы на них могут быть известны (supervised learning) или нет (unsupervised).

В нашем случае ситуация для применения алгоритмов машинного обучения оказалась очень благоприятной: основным результатом [8], помимо приложений, является программный пакет PyPHS ([9]), позволяющий проводить распределённые вычисления сложных систем, построенных в форме ПГС. Первая его часть (собственно построение) сводится к генерации систем уравнений, соответствующих ПГС с *заданной физикой и топологией взаимодействий*. То есть для задачи восстановления структуры ПГС в нашем распоряжении есть пакет, быстро и эффективно строящий базу данных с известным ответом – эта база используется в двух методах, описанных ниже.

Классический матричный подход

Чтобы свести задачу распознавания структуры ПГС к стандартной задаче машинного обучения, нужно выбрать формат представления данных. В данном случае цель – распределить правые части (2.2) по матрицам в (2.1). Технически для этого нужно выбрать некоторый базис функционального пространства, в котором мы будем работать. Самый простой выбор, естественно, рассмотреть многочлены от переменных, описывающих состояние системы. Он, конечно, не отражает всего разнообразия нелинейностей, возникающих в динамических системах, но для иллюстрации оказывается поучительным.

Таким образом, фаза обучения состоит в построении систем вида (2.1) пакетом PyPHS без дальнейшего их решения. Пакет (с открытым кодом) позволяет зафиксировать все параметры и ограничить пространство функций, а небольшая надстройка над ним обеспечивает построение нужного количества систем с заданным разбросом характеристик. Для нейронной сети входными параметрами будут: топология ПГС (узлы и связи между ними), коэффициенты всех многочленов, заменяющих функции в матрицах и в свёртках. Выходные данные – ана-

логичные коэффициенты разложения правых частей (2.2).

Для данной статьи мы начали с матриц с постоянными коэффициентами (то есть с примеров, соответствующих симплектическим структурам в узлах), а градиент Гамильтониана и другие функции ограничили квадратичными. Даже такой простой класс примеров оказался достаточно репрезентативным. Данные обрабатывались с помощью конволюционных нейронных сетей (convolution neural networks – некоторые подробности в приложении А).

Тестовый эксперимент проводился на небольшой выборке из 1500 элементов, для систем из 10 уравнений с возможной топологией, включающей 4, 5 или 6 рёбер. Даже для таких относительно небольших систем стало понятно, что выбор гиперпараметров играет важную роль, а количество параметров модели, необходимых для разумных результатов (потери при обучениии ~0.13, потери валидации ~0.007) достигает сотен тысяч. На стандартном компьютере (core i5, 2.6GHz) такой счёт занимает примерно 50 минут.

Основная проблема данного подхода в том, что для реальных систем (большого размера с сильными нелинейностями) количество параметров и время исполнения растёт очень быстро и требует либо огромных вычислительных мощностей, либо существенного упрощения модели. Таким образом, приведённый матричный подход в принципе работает как быстрое решение, не требующее существенных трудозатрат, но на практике он недостаточно удобен.

Машинное обучение на графах

Разумной альтернативой описанному выше матричному подходу оказался метод анализа связности графов и группировки вершин в них. Мы снова не пытаемся изложить всё множество алгоритмов анализа структуры графов, известных в современной литературе. Приведём лишь все необходимые “детали” для построения соответствующего алгоритма.

Как и раньше, у нас есть возможность построить базу данных для обучения с помощью пакета PyPHS, с абсолютно аналогичными входными и выходными данными. Но первый шаг алгоритма теперь существенно отличается: вместо заполнения матриц функциональными слагаемыми, мы сохраним лишь данные о взаимодействии переменных. Иными словами, на вход подаются не конретные функции из правых частей (2.2), а информация, от каких переменных (и как) они зависят. На выходе же тоже не требуются конкретные выражения для правых частей (2.1), а только их структура в смысле ПГС.

На языке графов это значит, что исходный строится явно по правым частям системы уравнений (2.2); все переменные X_i объявляются независимыми вершинами, их пары связаны ребром, если одна их них присутствует в правой части уравнения для другой, такие рёбра обычно будут направленными. Конечный граф состоит из объединённых групп вершин с большим количеством попарных связей (узлы ПГС), рёбра же (снова направленные) соответствуют портам. Переход от начального (большого) к конечному (существенно меньшему) графу как раз и осуществляется недетерминистскими методами.

У каждого специалиста по анализу данных есть свой любимый пакет для кластеризации точек или свёртки (pooling) графов/сетей – это, как правило, довольно простые нейронные сети с небольшим количеством слоёв. В нашем случае в основе конечного решения лежит пакет MapEquation ([10]). В остальном подход снова довольно стандартный и похож на предыдущий: обучить сеть на базе данных из PyPHS, проверить на элементах из выборки и вне выборки. Чтобы не перегружать текст техническими подробностями, мы не будем приводить здесь конкретных цифр и параметров. К тому же сравнивать параметры сетей, полученных в этом и предыдущем пункте, не имеет большого смысла, поскольку они решают несколько разные задачи. Анализ графов даёт исключительно представление об узлах ПГС, в то время как матричный подход начинает также решать (а иногда и решает полностью) задачу разметки, восстанавливая Гамильтонову структуру. Таким образом понятно, что подход с графами работает значительно быстрее.

Отметим лишь, что в отличие от предыдущего подхода, данный совершенно не меняется в зависимости от степени нелинейности системы. Он зависит исключительно от её размера, а именно от комбинации количества переменных и связей, что обеспечивает разумные свойства для масштабирования. Это и повлияло на конечный выбор данного подхода для решения задачи восстановления структуры ПГС.

Отметим также, что подход оказался очень гибким. Например, построение исходного графа просто по бинарному свойству наличия или отсутствия соответствующей переменной в правой части может быть уточнено: в зависимости от вида или сложности слагаемого можно присваивать некоторым или всем рёбрам нетривиальный вес. К тому же некоторые типы взаимодействий можно “насильно” относить к узлам или портам – это реализуется добавлением слоя в нейронную сеть.

4. ВМЕСТО ЗАКЛЮЧЕНИЯ

В этой статье мы продолжили изучение порт-Гамильтоновых систем, а именно описали важную часть алгоритма восстановления их структуры – разбиение системы на узлы. Основная цель работы была убедиться, что методы машинного обучения могут применяться для решения данной задачи с минимальными усилиями по адаптации готовых программных пакетов. Методом проб и ошибок мы установили что наиболее разумным в этом смысле подходом является анализ графа связности системы дифференциальных уравнений.

Напомним, что исходная мотивация для представления произвольной системы уравнений в виде ПГС была связана с последующим использованием геометрических интеграторов — численных методов, сохраняющих некоторую геометрическую структуру — для проведения компьютерных вычислений. Таким образом, подход потенциально позволяет расширить область применения как классических ([11, 12]), так и более продвинутых ([13, 14, 15]) интеграторов. Иными словами, реализация описанных алгоритмов позволила превратить теорию ПГС из абстрактной “игры обозначений” в часть большого пакета для моделирования сложных систем с использованием параллельных и распределенных вычислений.

Разработка и описание данного подхода также натолкнули нас на пару технических замечаний, которые мы хотим изучить более подробно в последующих работах.

Во-первых, в разделе о нейронных сетях мы привели некоторые численные значения потерь при обучении и валидации. Важно заметить, что в этой статистике учитывалось точное восстановление структуры исходной ПГС. Но замеченные “ошибки” не значат, что алгоритм сработал совсем неверно. На самом деле, возможно, была построена другая структура ПГС для заданной системы уравнений, что с точки зрения приложений для распределённых вычислений, тоже будет подходящим решением. Иными словами, вполне возможно, критерии “успеха” решения задачи можно существенно ослабить.

Во-вторых, на графах мы рассмотрели использование сетей, которые обучались на порт-Гамильтоновых системах, но концептуально этап обучения можно было бы совсем пропустить. В таком случае система всё равно разобьётся на узлы, которые, правда, могут быть малопригодны для восстановления Гамильтоновой структуры в смысле [4]. Возможно, тем не менее, искомое решение может быть достигнуто не полным переобучением сети, а лишь незначительной её модерацией добавлением одного “ПГС-слоя” или изменением оптимизируемой функции.

Мы также пришли к некоторому концептуальному замечанию, которое уже упомянули несколько раз: мысль анализировать ПГС с помощью стандартных негеометрических алгоритмов можно развернуть. Если на вход нейронной сети, обученной на ПГС, подать систему уравнений общего вида, на выходе будет некоторая структура, похожая на взаимодействие физических систем. Такое восстановление “скрытой физики” системы, с одной стороны, снова позволит применить методы распределённого вычисления. С другой стороны, его можно воспринимать как подход к классификации систем дифференциальных уравнений. Стандартные методы анализа обычно начинаются с линеаризации и рассмотрения добавочных слагаемых особого вида. Мы сами использовали похожий подход в разделе о матричных методах. Понятно, что они очень сильно зависят от выбора исходной системы координат. Вспомним задачи построения нормальных форм Гамильтоновых систем, когда к симплектическим структурам и простым Гамильтонианам добавляются нелинейные слагаемые с малыми параметрами. Если же посмотреть на более сложную систему с точки зрения ПГС, то вместо исходной простой Гамильтоновой системы, будет рассматриваться набор потенциально сильно нелинейных, но консервативных систем, возможно с совершенно не физической “энергией”. А “возмущение”, опять же не обязательно малое, будет соответствовать их взаимодействию.

ПРИЛОЖЕНИЕ А. НЕКОТОРЫЕ ТЕХНИЧЕСКИЕ ПОДРОБНОСТИ

Данные, описанные в матричном подходе секции 3, обрабатывались полной конволюционной нейронной сетью, состоящей из двух конволюционных слоёв, активационного (activation) и отбрасывающего (dropout) слоя, а также слоя нормализации выборки. В качестве оптимизационной была выбрана функция Адама, а функция потерь – среднеквадратичная.

Оптимизатор Адама корректирует веса по правилу:

$$w_t = w_{t-1} - \eta \frac{m_t}{\sqrt{\nu_t} + \varepsilon},$$

где w_t — веса t-го слоя, ν_t – параметр шага.

$$m_t = \frac{m_t}{1 - \beta_1^t},$$

$$\nu_t = \frac{\nu_t}{1-\beta_2^t},$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,$$

$$\nu_t = \beta_2 \nu_{t-1} + (1-\beta_2) g_t^2,$$

где g_t – градиент по текущей мини-выборке, β_t – гиперпараметры с исходнымии значениями, близкими к 1; переменные m_t, v_t суть среднее и смещённое отклонение градиентов функций потерь.

Количество фильтров для каждого конволюционного слоя было выбрано равным 64128. Количество “тренируемых” параметров равно 175162.

БЛАГОДАРНОСТИ

Я благодарен Всеволоду Сальникову за ценные советы по поводу машинного обучения и описание возможностей и проблем использования нейронных сетей. Некоторые предварительные численные эксперименты, мотивировавшие написание этой статьи, были выполнены Дарьей Лозиенко. Я благодарен Антуану Фалезу за многочисленные обсуждения возможностей пакета PyPHS. Я особенно благодарен Сергею Александровичу Абрамову за бесконечное терпение во время подготовки этой статьи и анонимному рецензенту за полезные замечания.

ИСТОЧНИКИ ФИНАНСИРОВАНИЯ

Данная работа была частично поддержана проектом CNRS PEPS Jeune Chercheur GraNum 2.0, а также проектом ACI Университета Ла-Рошели.

СПИСОК ЛИТЕРАТУРЫ

1. *Salnikov V., Hamdouni A., Loziienko D.*, Generalized and graded geometry for mechanics: a comprehensive introduction // Mathematics and Mechanics of Complex Systems. 2021. V. 9. № 1. 2021.
2. *Salnikov V., Hamdouni A.* Geometric integrators in mechanics: The need for computer algebra tools, Tr. Tret’ei Mezhdun. Konf. “Computer algebra” (Proc. 3rd Int. Conf. Computer Algebra), Moscow, 2019.
3. *Salnikov V.N., Hamdouni A.* Differential geometry and mechanics: A source for computer algebra problems // Program. Comput. Software. 2020. V. 46. P. 126–132.
4. *Salnikov V., Falaize A., Lozienko D.* Learning port-Hamiltonian systems: Algorithms // Comput. Math. Math. Phys. 2023. V. 63. P. 126–134.
5. *Paynter H.M.* Analysis and Design of Engineering Systems // MIT Press, Cambridge, Massachusetts, 1961.
6. *A. van der Schaft*. Port-Hamiltonian systems: an introductory survey // Proceedings of the International Congress of Mathematicians, Madrid, 2006.
7. Sage Manifolds – Differential geometry and tensor calculus with SageMath, https://sagemanifolds.obspm.fr
8. *Falaize A.* Modélisation, simulation, génération de code et correction de systèmes multi-physiques audios: Approche par réseau de composants et formulation hamiltonienne à ports, // PhD thesis, Télécommunication et Électronique de Paris, Université Pierre et Marie Curie, 2016.
9. Modeling, simulation and code-generation of multiphysical Port-Hamiltonian Systems in Python: https://github.com/pyphs/pyphs
10. *Edler D., Holmgren A. Rosvall M.*, Infomap – Network community detection using the MapEquation framework, https://www.mapequation.org/infomap/
11. *Hairer E., Lubich C., Wanner G.*, Geometric Numerical Integration // Springer Series in Computational Mathematics, 2006.
12. *Razafindralandy D., Hamdouni A., Chhay M.*, A review of some geometric integrators // Advanced Modeling and Simulation in Engineering Sciences, SpringerOpen. 2018. V. 5 № 1. P. 16.
13. *Razafindralandy D., Salnikov V., Hamdouni A., Deeb A.* Some robust integrators for large time dynamics // Advanced Modeling and Simulation in Engineering Sciences. 2019. V. 6. № 5.
14. *Cosserat O.*, Symplectic groupoids for Poisson integrators // Journal of Geometry and Physics, 2023. V. 186.
15. *Cosserat O., Laurent-Gengoux C., Salnikov V.* // Numerical Methods in Poisson Geometry and their Application to Mechanics, Preprint: arXiv:2303.15883.

PORT-HAMILTONIAN SYSTEM: STRUCTURE RECOGNITION AND APPLICATIONS

© 2024 V. N. Salnikov[a]

[a] *University of La Rochelle, Av. Michel Crépeau, La Rochelle, 17042, Paris, France*

In this paper, we continue to consider the problem of recovering the port-Hamiltonian structure for an arbitrary system of differential equations. We complement our previous study on this topic by explaining the choice of machine learning algorithms and discussing some details of their application. We also consider the possibility provided by this approach for a potentially new definition of canonical forms and classification of systems of differential equations.

Keywords: geometrization of mechanics, port-Hamiltonian systems, and machine learning methods for ODEs

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 100–107*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 004.421.6+512.644

НИЖНИЕ ГРАНИЦЫ ДЛЯ РАНГА МАТРИЦЫ С НУЛЯМИ И ЕДИНИЦАМИ ВНЕ ГЛАВНОЙ ДИАГОНАЛИ

© 2024 г. А. В. Селиверстов[a], * (ORCID: 0000-0003-4746-6396),
О. А. Зверков[a], ** (ORCID: 0000-0002-8546-364X)

[a] *Институт проблем передачи информации им. А.А. Харкевича РАН*
127051 Москва, Большой Каретный пер., д. 19, стр. 1, Россия

**E-mail: slvstv@iitp.ru*
***E-mail: zverkov@iitp.ru*

Поступила в редакцию 12.07.2023
После доработки 10.08.2023
Принята к публикации 01.10.2023

Найдена нижняя граница для ранга квадратной матрицы, у которой каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали каждый элемент равен либо нулю, либо единице. Ранг такой матрицы не меньше половины порядка матрицы. При дополнительном условии нижняя граница на единицу выше. Это условие означает отсутствие двоичного решения у некоторой вспомогательной системы линейных уравнений. Даны примеры, показывающие достижимость указанной нижней границы. Полученная нижняя граница для ранга позволяет свести задачу о поиске двоичного решения для системы линейных уравнений, в которой число линейно независимых уравнений достаточно велико, к аналогичной задаче от меньшего числа переменных. Найдены ограничения на существование большого множества решений, каждое из которых отличается от двоичного решения значением одной переменной. Отдельно обсуждается возможность для сертификации отсутствия двоичного решения у системы из большого числа линейных алгебраических уравнений. Также даны оценки времени работы для вычисления ранга матрицы в системе компьютерной алгебры SymPy. Показано, что ранг матрицы над полем вычетов по модулю простого числа вычисляется за меньшее время, чем обычно требуется для вычисления ранга матрицы того же порядка над полем рациональных чисел.

Ключевые слова: ранг матрицы, система линейных уравнений, система компьютерной алгебры, SymPy

DOI: 10.31857/S0132347424020133 **EDN:** RNNVQZ

1. ВВЕДЕНИЕ

Обозначим через K поле, вычислимое за полиномиальное время [1], характеристика которого равна либо нулю, либо нечётному простому числу. Решение системы из m уравнений от n переменных называется (0,1)-решением, если каждая переменная принимает одно из двух значений 0 или 1. Решение называется почти-(0,1)-решением, если одна переменная не равна ни 0, ни 1, а каждая из остальных равна 0 или 1.

Система линейных уравнений определяет подпространство в объемлющем аффинном пространстве с фиксированной системой декартовых координат. Мы отождествляем точки со списками элементов поля или со столбцами матрицы. Над неупорядоченным полем понятие многогранника не определено, но мы отождествляем множество вершин единичного куба в n-мерном пространстве с множеством из 2^n точек, координаты которых принадлежат множеству {0,1}. Две вершины этого куба, т.е. две (0,1)-точки, называются смежными, если они различаются по одной координате. Так, (0,1)-решение системы уравнений — вершина этого куба, принадлежащая данному подпространству; почти-(0,1)-решение — точка, лежащая на прямой, проходящей через две смежные вершины единичного куба, но не совпадающая с вершиной. Точка, все координаты которой равны 1/2, служит центром симметрии единичного куба.

Задача распознавания (0,1)-решения эквивалентна задаче о взаимном расположении подпространства и вершин единичного куба. Эта задача NP-полная. Используя оценки ранга матрицы специального вида, мы предлагаем необходимое условие существования достаточно большого набора почти-(0,1)-решений при отсутствии (0,1)-решений у системы уравнений. Существование почти-(0,1)-решений служит препятствием к снижению размерности задачи распознавания (0,1)-решений посредством исключения переменных, т.е. проектирования на координатное подпространство. Поэтому нарушение найденного нами условия означает возможность снижения размерности, следовательно, снижения

вычислительной сложности. Но мы не будем рассматривать задачи перечисления, которые труднее задач распознавания хотя бы одного решения [2, 3].

Недавно были предложены алгоритмы для распознавания (0,1)-решений системы линейных уравнений с целыми коэффициентами: как эвристические при условии малой плотности [4] или для достаточно большого числа уравнений [5, 6], так и недетерминированные с новыми верхними границами вычислительной сложности [7]. Наши новые результаты остаются справедливыми над конечными полями. При этом над конечным полем исключение переменных не сопровождается ростом длины записи коэффициентов уравнений. Поэтому многие вычисления относительно легко выполнить в системах компьютерной алгебры, а их битовая сложность близка к алгебраической сложности. Более того, над конечным полем при фиксированном числе переменных возможен полный перебор [8].

Ранг квадратной матрицы M связан с размерностью аффинной оболочки L точек, соответствующих столбцам матрицы. Если L содержит начало координат, то $\operatorname{rank}(M) = \dim(L)$, иначе $\operatorname{rank}(M) = \dim(L) + 1$.

Ранг $n \times n$ матрицы над полем можно вычислить, используя полиномиальное число процессоров и выполнив на каждом из них лишь $O(\log_2^2 n)$ операций над этим полем [9, 10]. С другой стороны, сложность вычисления ранга [11] и характеристического многочлена [12, 13] близка к сложности матричного умножения. Также для вычисления нормальной формы Смита целочисленной матрицы известен быстрый вероятностный алгоритм [14]. Вычисление ранга над кольцами без нетривиальных делителей нуля рассмотрено в работе [15]. Для разреженных симметричных матриц удобно использовать необходимое условие невырожденности, использующее многогранник Ньютона для квадратичной формы [16]. Многогранники Ньютона также полезны для решения других задач [17].

Однако на практике вычисление ранга матриц большого порядка требует больших затрат. Поэтому эффективно проверяемые оценки ранга могут быть полезны для различных приложений. Некоторые результаты о ранге матриц были апробированы на конференции по компьютерной алгебре, посвященной памяти Марко Петковшека (Marko Petkovšek) [18].

В разделе 2 представлены новые оценки ранга матрицы и связанные теоретические результаты. В разделе 3 обсуждаются результаты вычислений в системе компьютерной алгебры SymPy. В разделе 4 дано краткое заключение.

2. ТЕОРЕТИЧЕСКИЕ РЕЗУЛЬТАТЫ

Пусть система уравнений от n переменных имеет для каждого индекса $1 \leqslant k \leqslant n$ некоторое почти-(0,1)-решение, у которого значение координаты $x_k \notin \{0,1\}$. Такие решения соответствуют столбцам матрицы, у которой каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали равен либо нулю, либо единице. Оценка ранга такой матрицы позволяет оценить размерность аффинного подпространства, заданного исходной системой уравнений.

Теорема 1. *Дана $n \times n$ матрица M над полем K, в которой каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали равен либо нулю, либо единице. Ранг матрицы M не меньше числа $n/2$.*

Доказательство. Если $n = 2$, то ранг 2×2 матрицы M не меньше числа $n/2$, поскольку $\operatorname{rank}(M) \geqslant 1$.

Предположим, что для некоторого $n \geqslant 3$ теорема доказана для всех рассматриваемых $m \times m$ матриц порядка $m < n$. Рассмотрим $n \times n$ матрицу M.

Столбец матрицы M соответствует точке на прямой, проходящей через две смежные (0,1)-точки, но не совпадающей с этими (0,1)-точками. Аффинные преобразования вида $x_k \to 1 - x_k$ отображают (0,1)-точки в другие (0,1)-точки, а почти-(0,1)-точки в другие почти-(0,1)-точки. При этих преобразованиях сохраняется размерность подпространства. Преобразование $x_k \to 1 - x_k$ означает замену всех элементов в k-й строке матрицы. Это позволяет перейти от матрицы M к матрице $\widehat{M}$ того же типа, но в последнем столбце матрицы $\widehat{M}$ все элементы, кроме элемента на главной диагонали, равны нулю. Матрица

$$\hat{M} = \left(\begin{array}{ccc|c} & & & 0 \\ & N & & \vdots \\ & & & 0 \\ \hline * & \cdots & * & \alpha \end{array}\right)$$

для некоторого $\alpha \notin \{0,1\}$. Более того, выполнено неравенство $\operatorname{rank}(M) \geqslant \operatorname{rank}(\widehat{M}) - 1$. Но если аффинная оболочка столбцов матрицы M содержит начало координат, то ранг матрицы M может быть меньше ранга матрицы $\widehat{M}$.

Выполняя элементарные преобразования над столбцами матрицы M, получим матрицу

$$\widetilde{M} = \left(\begin{array}{ccc|c} & & & 0 \\ & N & & \vdots \\ & & & 0 \\ \hline 0 & \cdots & 0 & \alpha \end{array}\right)$$

того же ранга. У матриц M и $\widetilde{M}$ могут отличаться лишь нижние строки. В нижней строке матрицы $\widetilde{M}$ за исключением элемента на главной диагонали остальные элементы равны нулю.

Удаляя последний столбец и последнюю строку из матрицы $\widetilde{M}$, мы получим $(n-1)\times(n-1)$ матрицу N меньшего ранга. По предположению индукции, $\operatorname{rank}(N) \geqslant (n-1)/2$. Поэтому выполнено неравенство $\operatorname{rank}(\widetilde{M}) \geqslant n/2$.

Обозначим через L аффинную оболочку столбцов матрицы $\widetilde{M}$. Возможны два случая. Если L проходит через начало координат, то $\operatorname{rank}(\widetilde{M}) = \dim(L)$. В этом случае $\operatorname{rank}(M) \geqslant \dim(L) = \operatorname{rank}(\widetilde{M}) \geqslant n/2$.

Если L не проходит через начало координат, то $\operatorname{rank}(M) \geqslant \operatorname{rank}(\widetilde{M}) - 1 = \operatorname{rank}(N)$. Аффинная оболочка столбцов матрицы N не проходит через начало координат. Вновь применим преобразования вида $x_k \to 1 - x_k$ к матрице N и получим матрицу $\widetilde{N}$ того же типа, но в последнем столбце матрицы $\widetilde{N}$ все элементы, кроме элемента на главной диагонали, равны нулю. Более того, $\operatorname{rank}(N) \geqslant \operatorname{rank}(\widetilde{N})$. Удаляя из матрицы $\widetilde{N}$ последний столбец и последнюю строку, получим $(n-2)\times(n-2)$ матрицу U меньшего ранга. По предположению индукции её ранг ограничен снизу $\operatorname{rank}(U) \geqslant (n-2)/2$. Тогда $\operatorname{rank}(\widetilde{N}) = \operatorname{rank}(U) + 1 \geqslant n/2$. Следовательно, $\operatorname{rank}(M) \geqslant \operatorname{rank}(N) \geqslant \operatorname{rank}(\widetilde{N}) \geqslant n/2$.

Следующий результат показывает, что эта нижняя оценка ранга точная. При этом используется деление на два. Деление на два объясняет предположение, что характеристика поля K не равна двум. Обозначим через $\lceil\cdot\rceil$ результат округления до большего целого.

Теорема 2. *Для любого нечетного n существует такая $n\times n$ матрица M над полем K, что каждый элемент на главной диагонали отличен от нуля и от единицы, вне главной диагонали — равен либо нулю, либо единице, никакая* (0,1)*-точка не лежит в аффинной оболочке столбцов матрицы M и выполнено равенство* $\operatorname{rank}(M) = \lceil n/2 \rceil$.

Доказательство. Рассмотрим $n\times n$ матрицу

$$M = \begin{pmatrix} 1/2 & 0 & 1 & 0 & 1 & \cdots & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

и через N обозначим $(n-1)\times(n-1)$ подматрицу, получаемую удалением из матрицы M первого столбца и первой строки. Ранги матриц связаны равенством $\operatorname{rank}(M) = \operatorname{rank}(N) + 1$. Матрица N блочно-диагональная с 2×2 блоками, каждый блок вырожденный. Поэтому её ранг равен числу блоков: $\operatorname{rank}(N) = (n-1)/2$. Следовательно, $\operatorname{rank}(M) = \operatorname{rank}(N) + 1 = (n+1)/2 = \lceil n/2 \rceil$.

Каждый столбец матрицы M служит решением для системы из $(n+1)/2$ линейно независимых уравнений

$$\begin{cases} 2x_1 - x_2 - \cdots - x_{2k} - \cdots - x_{n-1} = 1 \\ x_{2k} + x_{2k+1} = 0,\ 1 \leqslant k \leqslant (n-1)/2. \end{cases}$$

Эта система не имеет (0,1)-решений. Действительно, из нижних уравнений следует, что (0,1)-решение должно бы иметь нулевые координаты, кроме первой. Но это несовместимо с первым уравнением.

Например, для 3×3 матрицы

$$\begin{pmatrix} 1/2 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

ранг равен двум. Три столбца соответствуют трем точкам на одной прямой L, которая задана системой из двух уравнений

$$\begin{cases} 1 - 2x_1 + x_2 = 0 \\ x_2 + x_3 = 0. \end{cases}$$

Эта система не имеет (0,1)-решений.

Для 4×4 матрицы

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 1/2 \end{pmatrix}$$

ранг равен трём. Четыре столбца соответствуют точкам на плоскости, заданной системой из двух уравнений

$$\begin{cases} x_3 = x_1 + x_2 + 1 \\ x_4 = (-x_1 + x_2 + 1)/2. \end{cases}$$

Эта система не имеет (0,1)-решений.

Для 2×2 матриц, в которых каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали равен либо нулю, либо единице, ранг равен единице лишь для матриц

$$\begin{pmatrix} 1/\alpha & 1 \\ 1 & \alpha \end{pmatrix},$$

где $\alpha \notin \{0,1\}$. Столбцы соответствуют точкам на прямой, проходящей через начало координат и заданной уравнением $x_2 = \alpha x_1$. Следовательно, если никакая (0,1)-точка не принадлежит прямой, проходящей через точки, соответствующие столбцам матрицы M, то $\operatorname{rank}(M) = 2$.

Теорема 3. *Даны четное число n и $n \times n$ матрица M над полем K, в которой каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали равен либо нулю, либо единице. Если никакая* (0,1)*-точка не лежит в аффинной оболочке столбцов матрицы M, то ранг матрицы M не меньше числа* $(n/2) + 1$.

Доказательство. Применение к строкам матрицы преобразований типа $x_k \to 1 - x_k$, как при доказательстве теоремы 1, позволяет перейти от матрицы M к матрице $\widehat{M}$ того же типа, но в последнем столбце матрицы $\widehat{M}$ все элементы, кроме элемента на главной диагонали, равны нулю. Матрица

$$\hat{M} = \left(\begin{array}{ccc|c} & & & 0 \\ & N & & \vdots \\ & & & 0 \\ \hline * & \cdots & * & \alpha \end{array}\right)$$

для некоторого $\alpha \notin \{0,1\}$. Поскольку аффинная оболочка столбцов матрицы M не содержит никакой (0,1)-точки, это верно и для матрицы $\widehat{M}$. При этом ранг не меняется. Поэтому $\operatorname{rank}(M) = \operatorname{rank}(M) = \operatorname{rank}(N) + 1$. По теореме 1, $\operatorname{rank}(N) \geqslant (n-1)/2$. Следовательно, выполнено неравенство $\operatorname{rank}(M) \geqslant (n+1)/2$. Для чётного n это неравенство эквивалентно неравенству $\operatorname{rank}(M) \geqslant (n/2) + 1$.

Теорема 4. *Дана система m линейно независимых линейных уравнений от n переменных, у которой нет* (0,1)*-решений. Если выполнено неравенство* $m > (n+1)/2$, *то для некоторого индекса* $1 \leqslant k \leqslant n$ *не существует почти-*(0,1)*-решения, в котором* $x_k \notin \{0,1\}$.

Доказательство. Пусть для каждой переменной существует почти-(0,1)-решение, отличающееся от нуля и от единицы значением этой переменной. Тогда легко составить матрицу M, в которой каждый элемент на главной диагонали отличен от нуля и от единицы, а вне главной диагонали равен либо нулю, либо единице. Поскольку нет (0,1)-решений, $\operatorname{rank}(M) = n - m + 1$. Если число n нечётное, то $n - m + 1 \geqslant n/2$ по теореме 1. Если число n чётное, то $n - m + 1 \geqslant (n/2) + 1$ по теореме 3. В любом случае $m \leqslant (n+1)/2$. Получено противоречие.

Обозначим через $\lfloor n/2 \rfloor$ целую часть числа $n/2$. Геометрическая интерпретация теоремы 4 заключается в следующем. Пусть $s < \lfloor n/2 \rfloor$. В n-мерном аффинном пространстве для каждого s-мерного подпространства L, которое не инцидентно никакой вершине единичного куба, существует забывающая координату проекция на некоторую координатную гиперплоскость, при которой образ подпространства L снова не инцидентен никакой вершине единичного куба. Проекция, забывающая выбранную координату, легко вычисляется. При этом таких проекций, вообще говоря, несколько, но выбор удачной проекции недетерминированный и может иметь высокую вычислительную сложность. В этом смысле обсуждаемый способ понижения размерности задачи напоминает результаты из работы [7]. Однако мы не используем вероятностных методов.

Теорема 5. *Дана система m линейно независимых линейных уравнений от n переменных над полем K. Если выполнены условия:* $n = 2m - 1$, *у этой системы нет* (0,1)*-решений, но для каждого индекса* $1 \leqslant k \leqslant n$ *существует почти-*(0,1)*-решение, в котором* $x_k \notin \{0,1\}$, *то точка* $(1/2, \ldots, 1/2)$, *каждая координата которой равна* 1/2, *не служит решением системы.*

Доказательство. Предположим, что точка $(1/2, \ldots, 1/2)$ служит решением системы. Тогда множество остальных решений этой системы распадается на пары симметричных решений, переходящих одно в другое при одновременном преобразовании всех координат $x_k \to 1 - x_k$. При этом преобразовании почти-(0,1)-решение переходит в другое почти-(0,1)-решение, у которого та же самая координата отличается от нуля и от единицы. Однако точка $(1/2, \ldots, 1/2)$ остаётся неподвижной.

Подстановкой нулевого значения вместо последней переменной $x_n = 0$ получим новую систему из m уравнений, у которой нет (0,1)-решений, но для каждого индекса $1 \leqslant k \leqslant n-1$ существует почти-(0,1)-решение, в котором значение $x_k \notin \{0,1\}$. В новой системе число линейно независимых уравнений равно m, а число переменных равно $n - 1 = 2m - 2$. Мы получаем противоречие с теоремой 4.

Следующий результат устанавливает взаимную зависимость почти-(0,1)-решений.

Теорема 6. *Если прямая L пересекает три прямые, каждая из которых содержит по две смежных* (0,1)*-точки, но прямая L не инцидентна никакой* (0,1)*-точке, то во всех точках на прямой L координаты, кроме некоторых трех координат, принимают какие-то постоянные значения из множества* {0,1}.

Доказательство. Без ограничения общности можно считать, что прямая L пересекает первую координатную ось в точке A с координатами $(\alpha, 0, ..., 0)$, у которой все координаты, кроме первой, равны нулю и $\alpha \notin \{0,1\}$. Для некоторого индекса $k \geqslant 2$ прямая L проходит через точку W, у которой все координаты, кроме k-й, принадлежат множеству $\{0,1\}$.

Прямая L состоит из точек $tA + (1-t)W$, где через t обозначен параметр. Если у точки W среди координат, кроме первой, какие-то две координаты равны единице, то эти координаты у любой третьей точки на прямой L тоже отличны и от нуля и от единицы. Однако по условию на прямой L найдётся третья точка, у которой ровно одна координата отлична от нуля и от единицы. Следовательно, у точки W не более трёх координат могут отличаться от нуля, включая первую. Поэтому прямая L лежит в координатном подпространстве размерности не выше трёх.

3. РЕАЛИЗАЦИЯ И ОБСУЖДЕНИЕ

Ранг матрицы быстрее вычисляется над полем $GF(p)$ вычетов по простому модулю p, чем над полем рациональных чисел, ср. [19]. Такие вычисления реализованы во многих системах компьютерной алгебры, например, в SymPy [20].

В системе SymPy 1.12 выполнены вычисления с матрицами, элементы которых независимо и равномерно распределены на конечном наборе значений. Для конечного поля — над множеством всех элементов поля. Матрицы генерировались методом randMatrix. Если время вычисления ранга одной матрицы меньше минуты, то это вычисление повторялось в пяти сериях. Тогда за время вычисления принимался минимум из пяти значений, каждое из которых получено усреднением внутри одной серии вычислений. Длина такой серии зависит от времени вычисления. Если время одного вычисления превышает две секунды, то каждая серия состоит из одного вычисления. Для каждого порядка матрицы вычислялась медиана по вычислениям для 25 матриц.

Вычисление ранга $n \times n$ матрицы над полем $GF(p)$ для $p \leqslant 11$ и $n \leqslant 500$ требует менее двух минут, а для $n \leqslant 1000$ требует менее 15 минут. При этом медиана времени вычисления ранга монотонно возрастает при увеличении порядка матрицы как $c(p)n^{3+\varepsilon}$, где в зависимости от простого числа p добавка в показателе степени меняется в интервале $0.05 < \varepsilon < 0.09$. Эта медиана монотонно возрастает при увеличении модуля p. Результаты вычислений для $p \in \{3,5,7,11\}$ показаны в табл. 1. Для $n = 500$ и при больших значениях $p \in \{31,101,307,1009,3001\}$ время вычисления ранга мало зависит от величины p. В табл. 2 для тех же данных показаны отношения межквартильного размаха $(Q_3 - Q_1)$ к медиане.

Таблица 1. Медиана времени в секундах для вычисления ранга случайной $n \times n$ матрицы над полем $GF(p)$ для $p \in \{3,5,7,11\}$

n	$GF(3)$	$GF(5)$	$GF(7)$	$GF(11)$
100	0.4	0.5	0.6	0.7
200	3.2	4.6	5.1	5.9
300	11	16	18	21
400	27	39	45	51
500	53	78	91	102
600	95	137	158	177
700	151	220	251	280
800	227	327	376	421
900	324	468	538	595
1000	447	644	737	832

Таблица 2. Отношения межквартильного размаха $(Q_3 - Q_1)$ к медиане времени для вычисления ранга случайной $n \times n$ матрицы над полем $GF(p)$ для $p \in \{3,5,7,11\}$

n	$GF(3)$	$GF(5)$	$GF(7)$	$GF(11)$
100	0.06	0.09	0.10	0.06
200	0.03	0.07	0.07	0.07
300	0.05	0.03	0.03	0.03
400	0.04	0.03	0.02	0.02
500	0.02	0.02	0.02	0.03
600	0.03	0.02	0.02	0.02
700	0.01	0.01	0.01	0.01
800	0.02	0.02	0.01	0.02
900	0.01	0.01	0.01	0.02
1000	0.02	0.01	0.01	0.01

Для матриц над полем рациональных чисел (в SymPy это домен QQ) время вычисления ранга быстрее возрастает при увеличении порядка матрицы, а также зависит от длины двоичной записи элементов матрицы. Генерировались матрицы, элементы которых независимо и равномерно распределены на множестве целых чисел от нуля до 10^k для значений показателя $k \in \{1,2,3,4,5\}$. При этом медиана времени вычисления ранга монотонно возрастает при увеличении порядка матрицы как $c(k)n^{4+\varepsilon}$, где в зависимости от k добавка в показателе степени меняется в интервале $0 < \varepsilon < 0.4$. Для $k = 1$ и $n = 1000$ измеренное время вычисления ранга $n \times n$ матрицы не превышает получаса, а для $k = 5$ — около трёх часов. Результаты приведены в табл. 3.

Таблица 3. Медиана времени в секундах для вычисления ранга случайной $n \times n$ матрицы с целочисленными элементами, которые независимо и равномерно распределены на отрезке от нуля до 10^k для $k \in \{1,2,3,4,5\}$

n	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
100	0.184	0.262	0.345	0.426	0.513
200	2.20	3.64	5.11	6.71	8.48
300	10.5	18.3	27.0	36.5	47.2
400	32.9	60.2	91.2	126	164
500	83.9	155	237	332	439
600	178	340	526	743	990
700	341	662	1040	1480	1980
800	609	1190	1880	2700	3630
900	1010	2000	3190	4600	6220
1000	1610	3200	5140	7440	10100

Вычисления проведены на персональном компьютере на базе процессора Intel® Core i7-5820K @ 3.30GHz и с оперативной памятью 32 гигабайта.

Обсуждаемое уменьшение числа переменных в задаче поиска (0,1)-решения можно пояснить через диалог между пользователем с малыми возможностями и веб-сервисом с большими возможностями для вычислений. Пользователь получает указания в виде сообщений небольшой длины, но хочет проверить наличие или отсутствие (0,1)-решения у системы линейных уравнений, не доверяя этому сервису. Если (0,1)-решение существует, то оно предъявляется и легко проверить, что указанная последовательность нулей и единиц служит решением. Если (0,1)-решения нет, то подсказка состоит в выборе переменных, которые можно исключить, чтобы новая система по-прежнему не имела (0,1)-решения. Исключение указанных переменных легко выполняется. По теореме 4 число переменных можно уменьшать, если исходная система имеет достаточно много линейно независимых уравнений. В результате система иногда сводится к одному уравнению. Если дальнейшее уменьшение числа переменных невозможно, то пользователю сообщают набор почти-(0,1)-решений. Тогда легко проверить, что нельзя дальше упростить систему. Теорема 2 показывает, что препятствия к упрощению существуют. В худшем случае задача остаётся вычислительно трудной.

4. ЗАКЛЮЧЕНИЕ

Изложенные результаты согласуются с общепринятой гипотезой о высокой вычислительной сложности задач распознавания (0,1)-решений для системы линейных уравнений, поскольку изменение задачи посредством исключения переменных встречает препятствие в худшем случае. Но полученные оценки оставляют возможность для некоторого понижения вычислительной сложности над конечными полями. С другой стороны, системы компьютерной алгебры позволяют быстро вычислять ранг матрицы и размерность аффинного подпространства над полем вычетов по простому модулю.

СПИСОК ЛИТЕРАТУРЫ

1. *Алаев П.Е.* Конечно порожденные структуры, вычислимые за полиномиальное время // Сибирский математический журнал. 2022. Т. 63. № 5. С. 953–974.
2. *Леонтьев В.К., Гордеев Э.Н.* О числе решений системы булевых уравнений // Автоматика и телемеханика. 2021. № 9. С. 150–168. DOI:10.31857/S0005231021090063
3. *Гордеев Э.Н., Леонтьев В.К.* О числе решений диофантова уравнения и проблеме Фробениуса // Журнал Вычислительной Математики и Математической физики, 2022. Т. 62. № 9. С. 1447–1457.
4. *Pan Y., Zhang F.* Solving low-density multiple subset sum problems with SVP oracle // Journal of Systems Science and Complexity. 2016. V. 29. P. 228–242. DOI:10.1007/s11424-015-3324-9
5. *Селиверстов А.В.* Двоичные решения для больших систем линейных уравнений // Прикладная Дискретная Математика. 2021. № 52. С. 5–15. DOI:10.17223/20710410/52/1
6. *Селиверстов А.В.* Обобщение задачи о сумме подмножеств и кубические формы // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 1. С. 51–60.
7. *Akmal S., Chen L., Jin C., Raj M., Williams R.* Improved Merlin–Arthur protocols for central problems in fine-grained complexity // Algorithmica. 2023. V. 85. P. 2395–2426. DOI:10.1007/s00453-023-01102-6
8. *Stoichev S.D., Gezek M.* Unitals in projective planes of order 25 // Mathematics in Computer Science. 2023. V. 17. No. 5. P. 1–19. DOI:10.1007/s11786-023-00556-9
9. *Chistov A.L.* Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic // In: L. Budach (eds) Fundamentals of Computation Theory. FCT 1985. Lecture Notes in Computer Science, vol. 199. Springer, Berlin, Heidelberg, 1985. P. 63–69. DOI:10.1007/BFb0028792
10. *Mulmuley K.* A fast parallel algorithm to compute the rank of a matrix over an arbitrary field // Combinatorica. 1987. V. 7. No. 1. P. 101–104. DOI:10.1007/BF02579205
11. *Cheung H.Y., Kwok T.C., Lau L.C.* Fast matrix rank algorithms and applications // Journal of the ACM.

2013. V. 60. No. 5. Article No. 31. P. 1–25. DOI:10.1145/2528404
12. *Переславцева О.Н.* О вычислении характеристического полинома матрицы // Дискретная математика. 2011. Т. 23. № 1. С. 28–45. DOI:10.4213/dm1128
13. *Neiger V., Pernet C.* Deterministic computation of the characteristic polynomial in the time of matrix multiplication // Journal of Complexity. 2021. V. 67. No. 101572. P. 1–35. DOI:10.1016/j.jco.2021.101572
14. *Birmpilis S., Labahn G., Storjohann A.* A fast algorithm for computing the Smith normal form with multipliers for a nonsingular integer matrix // Journal of Symbolic Computation. 2023. V. 116. P. 146–182. DOI:10.1016/j.jsc.2022.09.002
15. *Abramov S.A., Petkovšek M., Ryabenko A.A.* On ranks of matrices over noncommutative domains // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 5. С. 760–762.
16. *Юран А.* Многогранники Ньютона невырожденных квадратичных форм // Функциональный анализ и его приложения. 2022. Т. 56. № 2. С. 92–100. DOI:10.4213/faa3957
17. *Батхин А.Б., Брюно А.Д.* Вещественная нормальная форма бинарного многочлена в критической точке второго порядка // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 1. С. 3–15.
18. *Seliverstov A.V.* On a simple lower bound for the matrix rank // Компьютерная алгебра: материалы 5-й международной конференции. Москва, 26–28 июня 2023 г. / отв. ред. С.А. Абрамов, А.Б. Батхин, Л.А. Севастьянов. М.: ИПМ им. М.В. Келдыша, 2023. С. 126–128.
19. *Байрамов Р.Э., Блинков Ю.А., Левичев И.В., Малых М.Д., Мележик В.С.* Аналитическое исследование кубатурных формул на сфере в системах компьютерной алгебры // Журнал вычислительной математики и математической физики. 2023. Т. 63. № 1. С. 93–101.
20. *Meurer A., Smith C.P., Paprocki M., Čertik O., Kirpichev S.B., Rocklin M., Kumar A., Ivanov S., Moore J.K., Singh S., Rathnayake T., Vig S., Granger B.E., Muller R.P., Bonazzi F., Gupta H., Vats S., Johansson F., Pedregosa F., Curry M.J., Terrel A.R., Roučka Š., Saboo A., Fernando I., Kulal S., Cimrman R., Scopatz A.* SymPy: symbolic computing in Python // PeerJ Computer Science. 2017. V. 3. No. e103. P. 1–27. DOI:10.7717/peerj-cs.103

LOWER BOUNDS FOR THE RANK OF A MATRIX WITH ZEROS AND ONES OUTSIDE THE LEADING DIAGONAL

© 2024 A. V. Seliverstov[a], O. A. Zverkov[a]

[a] *Institute for Information Transmission Problems of the Russian Academy of Sciences (Kharkevich Institute), Bolshoy Karetny per. 19, build. 1, Moscow 127051 Russia*

We have found a lower bound on the rank of a square matrix, where every entry in the leading diagonal is neither zero nor one, but every entry outside the leading diagonal is either zero or one. The rank of such a matrix is at least half the order of the matrix. Under an additional condition, the lower bound is one higher. This condition means that some auxiliary system of linear equations has no binary solution. Examples are given showing the achievability of the lower bound. This lower bound on the rank allows us to reduce the problem of finding a binary solution to a system of linear equations, where the number of linearly independent equations is sufficiently large, to a similar problem in a smaller number of variables. Restrictions on the existence of a large set of solutions are found, each of which differs from binary one by the value of one variable. In addition, we discuss the possibility of certifying the absence of a binary solution to a system of a large set of linear algebraic equations. Estimates of the running time for calculating the rank of a matrix with the SymPy computer algebra system are also given. It is shown that the matrix rank over the field of residues modulo a prime number is calculated in less time than is usually required to calculate the rank of a matrix of the same order over the field of rational numbers.

Keywords: matrix rank, system of linear equations, computer algebra system, SymPy

REFERENCES

1. *Alaev P.E.* Finitely generated structures computable in polynomial time // Siberian Mathematical Journal. 2022. V. 63. № 5. P. 801–818. DOI:10.1134/S0037446622050019
2. *Leontiev V.K., Gordeev E.N.* On the number of solutions to a system of Boolean equations // Automation and Remote Control. 2021. V. 82. no. 9. P. 1581–1596. DOI:10.1134/S000511792109006X
3. *Gordeev E.N., Leont'ev V.K.* On the number of solutions to linear Diophantine equation and Frobenius problem. Computational Mathematics and Mathematical Physics. 2022. V. 62. № 9. P. 1413–1423. DOI:10.1134/S0965542522090044
4. *Pan Y., Zhang F.* Solving low-density multiple subset sum problems with SVP oracle // Journal of Systems Science and Complexity. 2016. V. 29. P. 228–242. DOI:10.1007/s11424-015-3324-9

5. *Seliverstov A.V.* Binary solutions to large systems of linear equations // Prikladnaya Diskretnaya Matematika. 2021. № 52. P. 5–15 (in Russian). DOI:10.17223/20710410/52/1
6. *Seliverstov A.V.* Generalization of the subset sum problem and cubic forms // Computational Mathematics and Mathematical Physics. 2023. V. 63. № 1. P. 48–56. DOI:10.1134/S0965542523010116
7. *Akmal S., Chen L., Jin C., Raj M., Williams R.* Improved Merlin–Arthur protocols for central problems in fine-grained complexity // Algorithmica. 2023. V. 85. P. 2395–2426. DOI:10.1007/s00453-023-01102-6
8. *Stoichev S.D., Gezek M.* Unitals in projective planes of order 25 // Mathematics in Computer Science. 2023. V. 17. № 5. P. 1–19. DOI:10.1007/s11786-023-00556-9
9. *Chistov A.L.* Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In: L. Budach (eds) Fundamentals of Computation Theory. FCT 1985. Lecture Notes in Computer Science, vol. 199. Springer, Berlin, Heidelberg, 1985. P. 63–69. DOI:10.1007/BFb0028792
10. *Mulmuley K.* A fast parallel algorithm to compute the rank of a matrix over an arbitrary field // Combinatorica. 1987. V. 7. № 1. P. 101–104. DOI:10.1007/BF02579205
11. *Cheung H.Y., Kwok T.C., Lau L.C.* Fast matrix rank algorithms and applications // Journal of the ACM. 2013. V. 60, no. 5. Article no. 31. P. 1–25. DOI:10.1145/2528404
12. *Pereslavtseva O.N.* Calculation of the characteristic polynomial of a matrix // Discrete Mathematics and Applications. 2011. V. 21. № 1. P. 109–128. DOI:10.1515/DMA.2011.008
13. *Neiger V., Pernet C.* Deterministic computation of the characteristic polynomial in the time of matrix multiplication // Journal of Complexity. 2021. V. 67. № 101572. P. 1–35. DOI:10.1016/j.jco.2021.101572
14. *Birmpilis S., Labahn G., Storjohann A.* A fast algorithm for computing the Smith normal form with multipliers for a nonsingular integer matrix // Journal of Symbolic Computation. 2023. V. 116. P. 146–182. DOI:10.1016/j.jsc.2022.09.002
15. *Abramov S.A., Petkovšek M., Ryabenko A.A.* On ranks of matrices over noncommutative domains // Computational Mathematics and Mathematical Physics. 2023. V. 63. № 5. P. 771–778. DOI:10.1134/S0965542523050020
16. *Yuran A.Y.* Newton polytopes of nondegenerate quadratic forms // Functional Analysis and Its Applications. 2022. V. 56. № 2. P. 152–158. DOI:10.1134/S0016266322020095
17. *Batkhin A.B., Bruno A.D.* Real normal form of a binary polynomial at a second-order critical point // Computational Mathematics and Mathematical Physics. 2023. V. 63. № 1. P. 1–13. DOI:10.1134/S0965542523010062
18. *Seliverstov A.V.* On a simple lower bound for the matrix rank. In: S.A. Abramov, A.B. Batkhin, L.A. Sevastyanov (eds.) Computer algebra: 5th International Conference Materials. Moscow, 26–28 June, 2023. Moscow: KIAM, 2023. P. 126–128.
19. *Bairamov R.E., Blinkov Yu.A., Levichev I.V., Malykh M.D., Melezhik V.S.* Analytical study of cubature formulas on a sphere in computer algebra systems // Computational Mathematics and Mathematical Physics. 2023. V. 63. № 1. P. 77–85. DOI:10.1134/S0965542523010050
20. *Meurer A., Smith C.P., Paprocki M., Čertik O., Kirpichev S.B., Rocklin M., Kumar A., Ivanov S., Moore J.K., Singh S., Rathnayake T., Vig S., Granger B.E., Muller R.P., Bonazzi F., Gupta H., Vats S., Johansson F., Pedregosa F., Curry M.J., Terrel A.R., Roučka Š., Saboo A., Fernando I., Kulal S., Cimrman R., Scopatz A.* SymPy: symbolic computing in Python // PeerJ Computer Science. 2017. V. 3. № e103. P. 1–27. DOI:10.7717/peerj-cs.103

ПРОГРАММИРОВАНИЕ, 2024, № 2, с. 108–117

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК. 004.421.6

ПОИСК ЛОРАНОВЫХ РЕШЕНИЙ УСЕЧЕННЫХ СИСТЕМ ЛИНЕЙНЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ EG-ИСКЛЮЧЕНИЙ

© 2024 г. А. А. Рябенко[a], *, Д. Е. Хмельнов[a], **

[a]*Федеральный исследовательский центр "Информатика и управление" РАН*
119333 Москва, ул. Вавилова, 40, Россия

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

Поступила в редакцию 31.08.2023
После доработки 10.09.2023
Принята к публикации 01.10.2023

Рассматриваются лорановы решения систем линейных обыкновенных дифференциальных уравнений с усеченными степенными рядами в роли коэффициентов. Лорановы ряды в решениях также усечены. В качестве средства для построения таких решений мы используем индуцированные рекуррентные системы и ранее предложили алгоритм для случая, когда индцуцированная рекуррентная система имеет невырожденную ведущую матрицу. Алгоритм находит для рядов в решениях максимально возможное число членов, инвариантных относительно любых продолжений усеченных коэффициентов исходной системы. Ниже мы представляем результаты по расширению применимости нашего алгоритма на случай, когда ведущая матрица вырождена, с привлечением алгоритма EG-исключений в качестве вспомогательного средства. Представлены реализация алгоритма в виде Maple-процедуры и примеры ее использования.

Ключевые слова: системы усеченных линейных обыкновенных дифференциальных уравнений, усеченные лорановы решения, система компьютерной алгебры Maple, алгоритм EG-исключений

DOI: 10.31857/S0132347424020145 **EDN:** RNLWQP

1. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ

Рассматриваются системы вида

$$A_r(x)\theta^r y(x) + A_{r-1}(x)\theta^{r-1} y(x) + \cdots + A_0(x) y(x) = 0, \quad (1)$$

где $y(x) = (y_1(x), y_2(x), \ldots, y_m(x))^T$ — вектор неизвестных, $A_r(x), \ldots, A_0(x)$ — $m \times m$-матрицы коэффициентов с элементами в виде степенных рядов по x над алгебраически замкнутым полем чисел, $\theta = x\dfrac{d}{dx}$.

Решение $y(x) = (y_1(x), \ldots, y_m(x))^T$ дифференциальной системы (1), компоненты которого являются формальными рядами Лорана, называется *лорановым решением*:

$$y(x) = \sum_{n=\nu}^{\infty} u(n) x^n, \quad (2)$$

где $\nu \in \mathbb{Z}$, $u(n) = (u_1(n), \ldots, u_m(n))^T$ — вектор-столбцы коэффициентов ряда Лорана для $n \in \mathbb{Z}_{\geqslant \nu}$, $u(\nu)$ — ненулевой вектор. Число ν называется *валюацией* ряда.

Валюацией уравнения с коэффициентами-рядами называется наименьшая из валюаций коэффициентов уравнения. Системы вида (1) легко привести к такому виду, что каждое ее уравнение будет иметь валюацию ноль. В дальнейшем рассматриваем именно такие системы.

1.1. Случай полностью заданных систем

Для любой системы (1) полного ранга, коэффициенты которой являются алгоритмически заданными рядами (то есть задан алгоритм, который вычисляет коэффициент любого члена x^s любого ряда в системе), в [1] предложен алгоритм нахождения всех ее лорановых решений в том смысле, что для решения может быть найдено любое наперед заданное число начальных членов ряда. Этот алгоритм основан на вычислении *индуцированной рекуррентной системы* $R(u) = 0$, которой удовлетворяет последовательность векторов $u(n)$ из (2). Индуцированная рекуррентная система строится с помощью преобразования

$$x \to E^{-1},\ \theta \to n, \quad (3)$$

примененного к исходной системе (1). E^{-1} обозначает оператор сдвига: $E^{-1}u(n) = u(n-1)$. Таким образом, $R = B_0(n) + B_{-1}(n)E^{-1} + B_{-2}(n)E^{-2} + \ldots$ и индуцированная система записывается как

$$\begin{pmatrix} 1+O(x) & x+O(x^2) & x+O(x^2) \\ x+O(x^2) & O(x) & x+O(x^2) \\ O(x) & x+O(x^2) & 1+x+O(x^2) \end{pmatrix} \theta y(x) +$$
$$+ \begin{pmatrix} -2+x+O(x^2) & O(1) & x+O(x^2) \\ x+O(x^2) & 1+x+O(x^2) & 1+x+O(x^2) \\ x+O(x^2) & x+O(x^2) & -2+x+O(x^2) \end{pmatrix} y(x) = 0.$$

Рис. 1

$$\begin{pmatrix} 1+O(x) & x+O(x^2) & x+O(x^2) \\ x+O(x^2) & \boldsymbol{O(x^2)} & x+O(x^2) \\ \boldsymbol{x+O(x^2)} & x+O(x^2) & 1+x+O(x^2) \end{pmatrix} \theta y(x) +$$
$$+ \begin{pmatrix} -2+x+O(x^2) & \boldsymbol{x+O(x^2)} & x+O(x^2) \\ x+O(x^2) & 1+x+O(x^2) & 1+x+O(x^2) \\ x+O(x^2) & x+O(x^2) & -2+x+O(x^2) \end{pmatrix} y(x) = 0.$$

Рис. 2

$$B_0(n)u(n) + B_{-1}(n)u(n-1) + \ldots = 0, \qquad (4)$$

где $u(n) = (u_1(n), \ldots, u_m(n))^T$ — вектор-столбец неизвестной последовательности, такой что $u_i(n) = 0$ для всех отрицательных n с достаточно большой величиной $|n|$, $i = 1, \ldots, m$; $B_0(n), B_{-1}(n), \ldots$ — матрицы полиномов от n; $B_0(n)$ — *ведущая матрица* (из того, что валюации уравнений системы (1) равны нулю, следует, что $B_0(n)$ — ненулевая матрица).

Если $B_0(n)$ невырождена, то мы можем рассматривать уравнение $\det B_0(n) = 0$ в качестве *определяющего уравнения* исходной дифференциальной системы: множество целых корней этого алгебраического уравнения содержит все возможные валюации лорановых решений системы (1). Это, в частности, дает возможность найти нижнюю границу $v_{\min}$ валюаций всех лорановых решений этой системы. Если $\det B_0(n) = 0$ не имеет целых корней, то эта система не имеет лорановых решений. Если $B_0(n)$ вырождена, то сначала применяется алгоритм $\mathrm{EG}^{\infty}_{\sigma}$ (это версия исходного алгоритма EG из [2], представленная в [1], для бесконечных рекуррентных систем) для того, чтобы преобразовать индуцированную рекуррентную систему к охватывающей ее рекуррентной системе $\overline{R}(u) = 0$ того же вида с невырожденной ведущей матрицей $\overline{B}_0(n)$, обеспечивая возможностью найти нижнюю границу $v_{\min}$ валюаций всех лорановых решений системы (1) и вычислить произвольное количество начальных элементов последовательности $u(n)$, $n \geqslant v_{\min}$, являющейся решением системы $\overline{R}(u) = 0$. Рекуррентная система $\overline{R}(u) = 0$, дополненная множеством линейных ограничений, которые также вычисляются алгоритмом $\mathrm{EG}^{\infty}_{\sigma}$, имеет то же самое множество решений, что и $R(u) = 0$.

1.2. Случай усеченных систем

В нашем текущем исследовании мы занимаемся случаем, когда ряды в коэффициентах системы представлены в усеченном виде, возможно, с разной степенью усечения для разных рядов. Мы называем такие системы *усеченными системами*. Каждый усеченный ряд представляется как

$$a(x) + O(x^{t+1}), \qquad (5)$$

где $a(x)$ — полином, целое $t \geqslant \deg a(x)$ — *степень усечения*. Мы говорим, что в (5) члены ряда до степени t *известны* (или *заданы*), а при $s > t$ — *неизвестны* (или *не заданы*).

Продолжение усеченного ряда — это ряд (возможно, также усеченный), начальные члены которого совпадают со всеми известными членами исходного усеченного ряда. В свою очередь, продолжение усеченного уравнения — это уравнение, коэффициенты которого являются продолжениями коэффициентов исходного уравнения, а продолжение системы уравнений — это система, уравнения которой являются продолжениями исходной системы.

Пример усеченной системы приведен на рис. 1, и пример одного из ее продолжений — на рис. 2. На рис. 2 выделены новые, по сравнению с системой на рис. 1, заданные члены.

Рассматриваемая нами задача состоит в том, чтобы для заданной системы (1), коэффициенты которой являются усеченными рядами вида (5), найти в компонентах $y_i(x)$ лорановых решений максимально возможное число начальных членов, инвариантных относительно любых продолжений этой системы. При этом будем интересоваться такими решениями, в которых хотя бы в одной компоненте есть хотя бы один ненулевой член.

Инвариантность членов усеченного решения $y(x)$ означает, что множество лорановых решений (возможно, также усеченных) любого продолжения заданной системы будет содержать решение, начальные члены в компонентах которого совпадают со всеми членами в компонентах $y(x)$. Максимальность же числа членов в решении $y(x)$ означает, что продолжение хотя бы на одно слагаемое хотя бы одной компоненты решения приведет к нарушению инвариантности, т.е. будет существовать такое продолжение заданной системы, у которого нет решения, начальные члены в компонентах которого совпадают с таким продолжением решения.

Для системы на рис. 1 примером такого решения является вектор $(x^2 + O(x^3), O(x^3), O(x^3))^T$. Множество всех решений этой системы можно записать с использованием произвольной постоянной:

$$\begin{pmatrix} cx^2 + O(x^3) \\ O(x^3) \\ O(x^3) \end{pmatrix}, \qquad (6)$$

где c может принимать любое числовое значение, кроме 0, поскольку при $c = 0$ мы получим вектор $(O(x^3), O(x^3), O(x^3))^T$, который формально можно назвать усечением нулевого ряда, но он не является его усечением с максимальным числом известных членов. В дальнейшем в этой работе под произвольными постоянными мы будем понимать, что они могут принимать любые такие численные значения, которые не меняют валюацию ни одной компоненты усеченного вектора, в который входят.

Так же, как в работе [3], для системы (1) множество W всех усеченных лорановых решений с максимальной степенью усечения будем представлять набором $\overline{W} = [w_1, \ldots, w_p]$ вектор-столбцов, компоненты которых являются усеченными лорановыми рядами с произвольными постоянными в коэффициентах. Каждый из столбцов имеет вид

$$w_i = \begin{pmatrix} \sum\limits_{n=v_{i1}}^{q_{i1}} C_{i1n} x^n + O(x^{q_{i1}+1}) \\ \cdots \\ \sum\limits_{n=v_{im}}^{q_{im}} C_{imn} x^n + O(x^{q_{im}+1}) \end{pmatrix},$$

где для $i = 1, \ldots, p$, $j = 1, \ldots, m$, $n = v_{ij}, \ldots, q_{ij}$ коэффициент C_{ijn} является линейной комбинацией произвольных постоянных, которые могут принимать все такие числовые значения, что $C_{ijv_{ij}} \neq 0$. Таким образом v_{ij} — валюация соответствующего ряда, q_{ij} — его степень усечения. Валюации компонент разных векторов в наборе $\overline{W}$ различаются хотя бы для одной компоненты: для любых i_1, i_2 $(i_1 \neq i_2)$ выполняется $(v_{i_1,1}, \ldots, v_{i_1,m}) \neq (v_{i_2,1}, \ldots, v_{i_2,m})$. Т.е. множество W усеченных решений разбито на подмножества с различными комбинациями валюаций компонент. Для одних и тех же компонент разных векторов набора $\overline{W}$ степень усечения может быть различной. Любому усеченному решению системы (1) соответствуют некоторые определенные значения произвольных постоянных в одном из векторов набора $\overline{W}$.

Для системы на рис. 2 множество всех усеченных лорановых решений выписано на рис. 3. При сравнении (6) и рис. 3 видим, что некоторые продолжения заданной системы могут иметь множества усеченных лорановых решений, для представления которых требуется использовать большее количество произвольных постоянных, чем требуется для представления множества усеченных решений исходной системы. В этом случае возможна ситуация, что некоторый инвариантный коэффициент найденного усеченного лоранового решения заданной системы совпадает с начальными членами усеченного решения некоторого продолжения заданной системы в том случае, когда дополнительная произвольная постоянная берется равной нулю, а при других значениях этой дополнительной произвольной постоянной этот коэффициент может иметь иное значение. Это также означает, что не любое усеченное решение продолжения заданной системы является

$$\left[\begin{pmatrix} c_1 x^2 + O(x^3) \\ c_2 x^2 + 2c_2 x^3 + O(x^4) \\ -c_2 x^2 - 3c_1 x^3 + O(x^4) \end{pmatrix}, \begin{pmatrix} c_1 x^2 + O(x^3) \\ O(x^4) \\ -3c_1 x^3 + O(x^4) \end{pmatrix}, \begin{pmatrix} O(x^4) \\ c_2 x^2 + 2c_2 x^3 + O(x^4) \\ -c_2 x^2 + O(x^4) \end{pmatrix} \right]$$

Рис. 3

продолжением усеченного решения заданной системы, несмотря на то, что любое усеченное решение заданной системы имеет продолжение среди решений любого продолжения заданной системы.

В отличие от случая полностью заданных систем, для усеченных систем невозможно построить решения произвольной степени усечения. Это утверждение доказано в [4] для частного случая скалярных уравнений ($m=1$). В [4] мы представили алгоритм, который находит лорановы решения для скалярных уравнений. Под лорановыми решениями усеченного уравнения и системы мы в дальнейшем понимаем усеченные лорановы решения с максимальным числом инвариантных членов. Мы использовали индуцированные рекуррентные системы и *литералы* как основу для нахождения этих решений. Литералы — символьные обозначения, используемые во время процесса построения решений, чтобы представить незаданные коэффициенты усеченных рядов, входящих в систему. Наш алгоритм для усеченных систем является модификацией алгоритма для систем с алгоритмически заданными коэффициентами. Основная идея — представить усеченный ряд (5) алгоритмически: алгоритм возвращает заданный коэффициент ряда для $s \leqslant t$ и возвращает литералы для $s > t$. Т.е. для системы (1) каждый элемент каждой матрицы $A_k(x)$ $(k = 0,1,\ldots,r)$ на пересечении i-й строки и j-го столбца $(i, j = 1,\ldots,m)$ представляется в виде

$$(A_k)_{ij}(x) = \sum_{s=0}^{\infty} \Xi_{kij}(s) x^s, \qquad (7)$$

где алгоритм Ξ_{kij} для целого s возвращает коэффициент усеченного ряда $(A_k)_{ij}(x)$, если s не превосходит степень усечения $(A_k)_{ij}(x)$, обозначим ее через t_{kij}. Иначе, Ξ_{kij} возвращает литерал $U_{[ij],[ks]}$.

В [3] мы применили подход из [4] к системам с $m > 1$ и предложили алгоритм построения лорановых решений систем для случая, когда определитель ведущей матрицы индуцированной рекуррентной системы не равен нулю (то есть эта ведущая матрица невырождена) и не содержит литералы.

2. РАСШИРЕНИЕ АЛГОРИТМА

Представляемые в этой работе новые результаты в исследовании поиска лорановых решений рассматриваемых систем связаны с использованием алгоритма $\mathrm{EG}_\sigma^\infty$ с целью расширить применимость алгоритма из работы [3] на системы, индуцированные рекуррентные системы которых имеют вырожденные ведущие матрицы. Мы продолжаем адаптацию алгоритма для систем с алгоритмически заданными коэффициентами с помощью представления усеченных рядов в алгоритмически заданном виде с привлечением литералов. Предварительная версия этих результатов была представлена в работе [5].

2.1. Случай без проблемных литералов

Алгоритм $\mathrm{EG}_\sigma^\infty$ заключается в последовательном повторении шагов редукции и сдвига, продолжающемся до тех пор, пока строки ведущей матрицы остаются линейно зависимыми. На этапе редукции вычисляются коэффициенты этой зависимости; после этого уравнение, соответствующее одной из зависимых строк, заменяется линейной комбинацией уравнений, вследствие чего строка ведущей матрицы, соответствующая получившемуся новому уравнению, становится нулевой. На этапе сдвига к этому новому уравнению применяется оператор сдвига $E : n \mapsto n+1$, в результате чего элементы строки ведущей матрицы, соответствующей сдвинутому уравнению, приобретают новые значения. При том предположении, что система имеет полный ранг, завершимость работы алгоритма гарантируется при использовании простого правила выбора заменяемого уравнения. Во время редукции также пополняется конечный набор линейных ограничений, каждое из которых содержит конечное число элементов последовательности (решения рекуррентной системы) и является линейной комбинацией этих элементов с постоянными коэффициентами. Каждое линейное ограничение соответствует целому корню полинома, который является коэффициентом линейной зависимости строки ведущей матрицы, соответствующей заменяемому уравнению (далее мы называем этот полином *полиномом ограничений*).

Основное препятствие в использовании алгоритма $\mathrm{EG}_\sigma^\infty$ в случае усеченных систем в том, что в промежуточных вычислениях могут появиться *проблемные литералы* — литералы в проблемных местах, а именно в определителе ведущей матрицы или в полиномах ограничений, то есть в полиномах, у которых требуется находить все целые корни для дальнейших вычислений. Если после применения $\mathrm{EG}_\sigma^\infty$ литералов нет ни в определителе ведущей матрицы, ни в полиномах ограничений, то последующие вычисления с помощью полученной охватывающей рекуррентной системы, дополненной линейными ограничениями, аналогичные вычислениям в алгоритме из работы [3], дают все лорановы решения исходной дифференциальной системы, в чем и заключается предлагаемая расширенная версия алгоритма для этого случая без проблемных литералов.

Рассмотрим следующую систему:

$$\begin{pmatrix} 3x + O(x^2) & 7x^2 + O(x^4) \\ O(x^2) & 17x^2 + O(x^4) \end{pmatrix} \theta^2 y(x) +$$
$$+ \begin{pmatrix} -1 + 2x + O(x^2) & x + 5x^2 + O(x^4) \\ O(x^2) & 11x^2 + O(x^4) \end{pmatrix} \theta y(x) +$$
$$+ \begin{pmatrix} O(1) & x - 3x^2 + O(x^4) \\ 1 + O(x^2) & -6x^2 + O(x^4) \end{pmatrix} y(x) = 0. \quad (8)$$

Ведущая матрица ее индуцированной рекуррентной системы вырождена:

$$\begin{pmatrix} U_{[1,1],[0,0]} - n & 0 \\ 1 & 0 \end{pmatrix}.$$

$U_{[1,1],[0,0]}$ — литерал, обозначающий незаданный коэффициент при x^0 в (1,1)-м элементе матричного коэффициента $A_0(x)$ системы (8). После применения $\mathrm{EG}_\sigma^\infty$ преобразованная ведущая матрица охватывающей системы становится невырожденной:

$$\begin{pmatrix} 3n^2 + 2n + U_{[1,1],[0,1]} & n+1 \\ 1 & 0 \end{pmatrix},$$

и ее определитель $(-n-1)$ не содержит литералы. Применение $\mathrm{EG}_\sigma^\infty$ не привело к построению дополнительных линейных ограничений, соответственно, отсутствуют полиномы ограничений с литералами. Это случай, когда применим наш новый алгоритм и он строит все лорановы решения (c — произвольная постоянная):

$$\begin{pmatrix} 6cx^2 + O(x^3) \\ \dfrac{c}{x} + c + O(x) \end{pmatrix}.$$

2.2. Проблемные литералы

Рассмотрим другую систему:

$$\begin{pmatrix} O(x^5) & -1 + O(x^5) \\ 1 + O(x^5) & O(x^5) \end{pmatrix} \theta y(x) +$$
$$+ \begin{pmatrix} O(x^5) & O(1) \\ 2 + O(x^5) & O(x^5) \end{pmatrix} y(x). \quad (9)$$

Ведущая матрица ее индуцированной рекуррентной системы невырождена:

$$\begin{pmatrix} 0 & U_{[1,2],[0,0]} - n \\ 2 + n & 0 \end{pmatrix}.$$

Однако, ее определитель $(n - U_{[1,2],[0,0]})(2 + n)$ содержит литерал. Видно, что определитель имеет корни -2 и $U_{[1,2],[0,0]}$. Это означает, что множество целых корней этого определителя может быть различно для различных целых значений литерала $U_{[1,2],[0,0]}$. Легко проверить, что не существует лорановых решений этой системы. Это можно сделать, построив различные продолжения исходной системы, подставив различные значения литерала $U_{[1,2],[0,0]}$, и затем найдя лорановы решения этих продолжений с помощью нашего алгоритма из работы [3] (этот алгоритм применим к этим продолжениям, поскольку ведущие матрицы индуцированных рекуррентных систем для каждого из этих продолжений будут по прежнему невырожденными и не будут содержать литералы в их определителях). Например, для $U_{[1,2],[0,0]} = 5$ решением продолжения будет

$$\begin{pmatrix} O(x^{10}) \\ c_1 x^5 + O(x^6) \end{pmatrix},$$

а для $U_{[1,2],[0,0]} = 6$ решением продолжения будет

$$\begin{pmatrix} O(x^{11}) \\ c_1 x^6 + O(x^7) \end{pmatrix}.$$

Поскольку решения продолжений не имеют совпадающих начальных членов, то не существует лоранового решения исходной системы.

Согласно (3) и (7), для каждого коэффициента $B_{-s}(n)$ ($s = 0, 1, \ldots$) индуцированной системы (4) его элементы на пересечении i-й строки и j-го столбца ($i, j = 1, \ldots, m$) имеют вид

$$(B_{-s})_{ij}(n) = \sum_{k=0}^{r} \Xi_{kij}(s)(n-s)^k,$$

т.е. $(B_{-s})_{ij}(n)$ — полином от n, коэффициенты которого являются полиномами от литералов. Определитель ведущей матрицы индуцированной системы (4) также является полиномом от n и литералов. Если он равен нулю, то применяем редукции в алгоритме $\mathrm{EG}_\sigma^\infty$ таким образом, чтобы в охватывающей системе $\overline{R}(u) = 0$ все элементы всех коэффициентов $\overline{B}_{-s}(n)$ оставались полиномами от литералов.

Пусть $p(n)$ является определителем ведущей матрицы (либо индуцированной рекуррентной системы, если ее ведущая матрица невырожденная, либо охватывающей рекуррентной системы после применения $\mathrm{EG}_\sigma^\infty$ в противном случае). Если $p(n)$ содержит литералы, то в общем случае он может быть представлен в виде

$$p(n) = p_1(U_1,\ldots,U_\varsigma)\, p_2(n)\, p_3(n, U_1,\ldots,U_\varsigma), \quad (10)$$

где $U_1,\ldots,U_\varsigma$ — литералы, входящие в $p(n)$; $p_1(U_1,\ldots,U_\varsigma)$ не зависит от n и имеет максимально

возможную для $p(n)$ степень; $p_2(n)$ — полином от n, независящий от литералов, также максимально возможной для $p(n)$ степени; $p_3(n,U_1,\ldots,U_\varsigma)$ — полином от n с коэффициентами, полиномиально зависящими от литералов. Если $p_1(U_1,\ldots,U_\varsigma)$ существенно зависит от литералов (то есть не является числом), то существуют такие числовые значения $\alpha_1,\ldots,\alpha_\varsigma$, что $p_1(\alpha_1,\ldots,\alpha_\varsigma)=0$ и, следовательно, $p(n)=0$ для любого n, то есть ведущая матрица для этих значений литералов является вырожденной. Если $p_3(n,U_1,\ldots,U_\varsigma)$ не является числом, то решение алгебраического уравнения $p_3(N,U_1,\ldots,U_\varsigma)=0$ относительно $U_1,\ldots,U_\varsigma$ позволяет получить такие числовые значения $\alpha_1,\ldots,\alpha_\varsigma$, что $p_3(n,\alpha_1,\ldots,\alpha_\varsigma)$ имеет корень N для любого N, кроме не более чем конечного числа значений, при которых $p_3(N,U_1,\ldots,U_\varsigma)$ не зависит от литералов. Это означает, что в любом случае множество корней $p(n)$ не инвариантно по отношению к продолжениям заданной дифференциальной системы. Также это рассуждение показывает, что во всех случаях можно подобрать такие значения литералов, что максимальный целый корень определяющего уравнения равен любому достаточно большому заданному целому числу N. Аналогичное рассуждение применимо и для полиномов ограничений.

Система (9) является примером такого случая при том, что алгоритм $\mathrm{EG}_\sigma^\infty$ не применяется — для любого продолжения исходной системы индуцированная система имеет невырожденную ведущую матрицу. Верно следующее

Предложение. *Пусть дана усеченная система* (1) *такая, что определитель ведущей матрицы ее индуцированной системы* (4) *зависит от литералов. Пусть* $\det B_0(n)=p(n)$ *имеет такое представление* (10)*, что* $p_3(n,U_1,\ldots,U_\varsigma)$ *не является числом. Тогда система* (1) *не имеет лорановых решений.*

Доказательство. Как сказано выше, при выполнении условий предложения для любого достаточно большого целого N существуют числовые значения $\alpha_1,\ldots,\alpha_\varsigma$ такие, что максимальный целый корень полинома $p(n,\alpha_1,\ldots,\alpha_\varsigma)$ равен N.

Заметим, что присвоение числовых значений литералам — $U_i=\alpha_i$, $i=1,\ldots,\varsigma$, — соответствует новой системе с усеченными коэффициентами, которая является продолжением исходной системы. При этом определяющее уравнение новой системы — $p(n,\alpha_1,\ldots,\alpha_\varsigma)=0$ — не зависит от литералов. Построение лорановых решений для таких систем рассмотрено в [3]. В частности, там показано, что если ведущая матрица индуцированной системы невырождена и определяющее уравнение не зависит от литералов, то

- система имеет лорановы решения с валюацией v, только если v является целым корнем определяющего уравнения;
- система имеет по крайней мере лорановы решения с валюацией, равной максимальному целому корню определяющего уравнения.

Рассмотрим сначала случай, когда в разложении (10) множитель $p_2(n)$ не имеет целых корней, т.е. не существует целого v такого, что $p(v,U_1,\ldots,U_\varsigma)=0$ при любых значениях литералов. Т.е. $\det B_0(n)$ как полином от n с коэффициентами, полиномиально зависящими от литералов, не имеет целых корней. Тогда существуют числовые значения $\beta_1,\ldots,\beta_\varsigma$ такие, что полином $p(n,\beta_1,\ldots,\beta_\varsigma)$ не имеет целых корней. В этом случае по определению исходная система (1) не имеет лорановых решений, поскольку у нее есть как продолжения, соответствующие значениям $\alpha_1,\ldots,\alpha_\varsigma$, имеющие лорановы решения любой достаточно большой валюации, так и продолжения, соответствующие значениям $\beta_1,\ldots,\beta_\varsigma$, лорановых решений не имеющие.

Пусть теперь $p_2(n)$ имеет целые корни. Покажем, что для любого $v\in\mathbb{Z}$ такого, что $p_2(v)=0$, система (1) не имеет лорановых решений с валюацией v.

Система (1) имеет решение вида (2) с валюацией v, только если система

$$\begin{gathered}B_0(v)u(v)=0,\\ B_0(v+1)u(v+1)+B_{-1}(v+1)u(v)=0,\\ \ldots\\ B_0(N)u(N)+B_{-1}(N)u(N-1)+\ldots+\\ +B_{-N+v-1}(N)u(v+1)+B_{-N+v}(N)u(v)=0\end{gathered}\qquad(11)$$

алгебраических уравнений относительно неизвестных $u_1(v),\ldots,u_m(v)$, $u_1(v+1),\ldots,u_m(v+1),\ldots$, $u_1(N),\ldots,u_m(N)$ для любого $N\in\mathbb{Z}_{\geqslant v}$ имеет такое решение, что вектор $u(v)=(u_1(v),\ldots,u_m(v))^T$ — ненулевой.

Обозначим через $t_{\max}$ максимальное значение всех степеней усечения коэффициентов системы (1). Выберем числа $\alpha_1,\ldots,\alpha_\varsigma$ так, чтобы максимальный целый корень полинома $p(n,\alpha_1,\ldots,\alpha_\varsigma)$ стал равен $N>v+t_{\max}$. Рассмотрим систему (11) для соответствующей подстановки $U_i=\alpha_i$ системы-продолжения.

Поскольку $N>v+t_{\max}$, получаем, что каждый элемент матрицы $B_{-N+v}(N)$ на пересечении i-й строки и j-го столбца равен

$$(B_{-N+v})_{ij}(N) = \sum_{k=0}^{r} U_{[ij],[k,N-v]} v^k,$$

т.е. каждый элемент матрицы $B_{-N+v}(N)$ зависит от литералов $U_{[ij],[k,N-v]}$ ($k = 0,1,\ldots,r$, если $v \neq 0$, и $k=0$, иначе). Все остальные слагаемые в (11) — $B_0(v)u(v)$, $B_0(v+1)u(v+1),\ldots, B_{-N+v-1}(N)u(v+1)$ — именно от этих литералов не зависят, т.е. возможно и зависят от литералов, но таких, что их четвертый индекс меньше, чем $N-v$. Если вектор $u(v)$ ненулевой, то $B_{-N+v}(N)u(v)$ — это вектор, каждый элемент которого содержит литералы $U_{[ij],[k,N-v_{\max}]}$, причем i-й элемент зависит только от литералов с первым индексом i. Т.е. каждый элемент этого вектора зависит от тех литералов, от которых другие элементы вектора не зависят. Это верно и про вектор $V = -(B_{-1}(N)u(N-1)+\ldots+B_{-N+v}(N)u(v))$. Тогда последняя строка в (11) может быть переписана в виде $B_0(N)u(N) = V$, где $B_0(N)$ — вырожденная матрица по выбору $\alpha_1,\ldots, \alpha_\varsigma$ и V — вектор, все элементы которого — линейно независимые над полем чисел полиномы от литералов. Можно подобрать такие значения для литералов, входящих в V, что система $B_0(N)u(N) = V$ окажется несовместной, если $u(v)$ — ненулевой вектор. Следовательно, построенное для исходной системы продолжение не имеет лорановых решений с валюацией v, откуда по определению следует, что исходная система не имеет лорановых решений с валюацией v, из чего следует утверждение предложения. □

2.3. Вариативность EG_σ^∞

Алгоритм EG_σ^∞ может иметь вариативность в выполнении. Несмотря на то, что для выбора заменяемого уравнения нужно использовать специальное правило, чтобы гарантировать завершимость работы алгоритма, все равно может быть более одного варианта для выбора уравнения, которое будет заменено на шаге редукции. Это приводит к тому, что для одной и той же рекуррентной системы применение алгоритма EG_σ^∞ может строить разные охватывающие системы. Например, для системы (8) применение EG_σ^∞ может быть выполнено иным способом, которым строится охватывающая система с другой ведущей матрицей:

$$\begin{pmatrix} U_{[1,1],[0,0]} - n & 0 \\ 3n^2 + 2n + U_{[1,1],[0,1]} & n+1 \end{pmatrix},$$

определитель которой равен $(U_{[1,1],[0,0]} - n)(n+1)$ и содержит литерал $U_{[1,1],[0,0]}$. Легко видеть, что этот определитель очень похож на определитель ведущей матрицы индуцированной рекуррентной системы для системы (9), которая не имеет лорановых решений. Этот второй вариант выполнения алгоритма EG_σ^∞ дополнительно строит полином ограничений $-U_{[[1,1],[0,0]]} + n$, который также содержит этот литерал. При этом, из первого варианта выполнения алгоритма EG_σ^∞ мы знаем, что система (8) имеет лорановы решения. Это дает нам контр-пример к возможной гипотезе, что лорановых решений не существует всегда, когда определитель ведущей матрицы и/или полиномы ограничений содержат литералы после применения EG_σ^∞. Таким образом и в случае проблемных литералов возможно существование усеченных лорановых решений, причем этот случай может возникать или не возникать в зависимости от варианта вычисления в ходе применения алгоритма EG_σ^∞.

2.4. Экспериментальная версия алгоритма для случая проблемных литералов

Мы разработали экспериментальную версию нашего алгоритма для случая, когда определяющее уравнение и/или полиномы ограничений содержат литералы. Эта версия принимает во внимание только инвариантные целые корни определяющего уравнения и полиномов ограничений (то есть корней, которые не зависят от литералов). Дополнительно, по сравнению с вариантом алгоритма из работы [3], внесено еще несколько модификаций в процедуру вычисления решения индуцированной системы (или охватывающей индуцированной системы) и дополнительных линейных ограничений при их наличии:

• **Формирование U-ограничений.** Как и в работе [3], вычисления с помощью индуцированной (или ее охватывающей) рекуррентной системы проводятся, начиная с наименьшего целого корня определяющего уравнения, подставляя соответствующее n в рекуррентную систему и далее последовательно увеличивая значения n. Для каждого значения n из каждого уравнения рекуррентной системы выражаются еще невычисленные коэффициенты компонент решения через другие коэффициенты компонент решения, и некоторые из них в итоге останутся невычисленными и станут произвольными постоянными. Аналогично учитываются и линейные ограничения, если они есть. U-ограничения содержат пары $[a,b]$ из возникающих в ходе таких вычислений выражений вида $a+bs=0$ (такое выражение — либо некоторое уравнение рекуррентной системы, либо конкретное линейное ограничение), где s — еще не вычисленный коэффициент какой-то компоненты решения в промежуточных вычислениях,

b — коэффициент при s, возможно, содержащий литералы, a — линейная комбинация отличных от s еще не вычисленных коэффициентов компонент решений рекуррентной системы в промежуточных вычислениях, также, возможно, содержащая литералы. Пара $[a,b]$ добавляется в U-ограничение, если b содержит литералы. Ранее в работе [3] в этом случае не производилось выражение коэффициента $s=-a/b$, а в новой экспериментальной версии это делается, но при этом добавляется новая пара в $[a,b]$ в U-ограничения. Например, пусть в ходе вычислений возникло соотношение

$$U_{[[1,1],[0,0]]}u_1(2) - 6U_{[[1,1],[0,0]]}u_2(-1) - \\ - 2u_1(2) + 12u_2(-1) = 0,$$

где $u_i(n)$ — i-я компонента вектора $u(n)$. В качестве s выбирается $u_1(2)$, и в U-ограничения добавляется пара $[-6U_{[[1,1],[0,0]]}u_2(-1) + 12u_2(-1), U_{[[1,1],[0,0]]} - 2]$.

• **Освобождение от деления на выражение с литералами.** Если после вычисления с помощью рекуррентной системы и линейных ограничений оказывается, что произвольная постоянная в вычисленном решении делится на выражение с литералами (например, $c/(d(U))$, где $d(U)$ — полином от литералов), то она меняется на произвольную постоянную без такого деления ($c' = c/(d(U))$).

• **Проверка соответствия U-ограничениям.** Далее в ранее сформированные U-ограничения в парах $[a,b]$ делается замена входящих в них коэффициентов решений, которые были еще не вычисленными на момент формирования пар U-ограничений, на их уже вычисленные значения. Затем для каждой пары $[a,b]$ из получившегося соотношения $b=0$ выражаются литералы и они подставляются в a, и если при этом получается $a=0$, значит, данная пара инвариантна относительно любых продолжений исходной дифференциальной системы (так как, и при значениях литералов, при которых $b=0$, соотношение $a+bs=0$ всегда выполнено). Если же $a\neq 0$, то уравнение $a=0$ добавляется в систему алгебраических уравнений относительно произвольных постоянных, и далее эта система решается, в результате чего часть произвольных постоянных получает выражение через другие произвольные постоянные или обнуляется, что в итоге приводит к формированию итогового ответа, инвариантного относительно продолжений исходной усеченной дифференциальной системы.

Эксперименты показали, что такая версия алгоритма дает верные ответы для системы (8) при любых вариациях выполнения алгоритма $\mathrm{EG}_{\sigma}^{\infty}$ и для системы (9), а также для ряда других тестовых систем. Системы, для которых эта версиях давала бы неверные решения, в экспериментах не были обнаружены. В дальнейших исследованиях мы планируем либо доказать, что этот подход всегда корректен, либо идентифицировать ограничения его применимости.

3. РЕАЛИЗАЦИЯ И ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ

Реализация новой версии алгоритма для случая без проблемных литералов и экспериментальной версии для случая проблемных литералов, выполнена в среде системы компьютерной алгебры Maple 2023[1] ([6]) в виде обновленной версии процедуры LaurentSolution пакета TruncatedSeries [7, 8] (можно посетить веб-страницу

http://www.ccas.ru/ca/TruncatedSeries

для получения библиотеки с новой версией процедуры, примеров ее использования и другой дополнительной информации о пакете). Предыдущая версия процедуры реализовывала алгоритм для случая, когда ведущая матрица индуцированной рекуррентной системы была невырожденной и ее определитель не содержал литералы, и была представлена в работе [3]. Новая версия использует те же самые аргументы и формат результата.

Первый обязательный аргумент процедуры — система уравнений вида (1), записанная аналогично математической записи: применение θ^k к вектор-столбцу неизвестных $y(x)$ записывается как **theta(y(x),x,k)**, матричные коэффициенты задаются в виде стандартной структуры **Matrix** с элементами в виде выражений $a(x) + O(x^{t+1})$, где $a(x)$ — полином степени не выше t над полем алгебраических чисел, умножение матрицы на столбец задается с помощью точки, как это принято в среде Maple. Второй обязательный аргумент — имя, обозначающее вектор-столбец неизвестных, например, $y(x)$. Результат работы процедуры — множество усеченных лорановых решений, представленное в виде списка вектор-столбцов из набора $\bar{W}$: каждый вектор-столбец представлен списком компонент, каждый элемент этого списка имеет вид $C_v x^v + C_{v+1}x^{v+1} + \ldots + C_q x^q + O(x^{q+1})$, где v — валюация данной компоненты, q — степень усечения, C_n — вычисленные коэффициенты лоранова ряда, которые являются линейными комбинациями произвольных постоянных вида $\_c_1, \_c_2, \ldots$ Если в качестве результата выдается

[1] Работоспособность проверена также в версиях Maple 2022 и Maple 2021, в более ранних версиях работоспособность также возможна, но не проверялась и не гарантируется.

пустое множество, значит, лорановых решений не существует ни при каком продолжении заданной системы (определитель ведущей матрицы индуцированной системы не имеет целых корней). Если результат выдается в виде константы FAIL, то значит исходная система не имеет лорановых решений, хотя некоторые ее продолжения могут иметь такие решения.

Дополнительно нами была реализована вспомогательная процедура Substitute, которая принимает на вход решение (или предполагаемое решение) системы в том виде, в котором выдается результат процедуры LaurentSolution, а также систему и имя, обозначающее вектор-столбец неизвестных в системе (аналогично аргументам процедуры LaurentSolution). Процедура подставляет в систему каждый из вектор-столбцов из набора $\bar{W}$ в решении и выдает результаты подстановки в виде усеченных рядов. Также в качестве первого аргумента возможна передача не набора $\bar{W}$, а только одного из векторов.

Приведем примеры вычисления лорановых решений усеченных систем в системе Maple 2023. Систему (8) задаем следующим образом:

```
> S1:=Matrix([[3*x+O(x^2),7*x^2+O(x^4)],
              [O(x^2),17*x^2+O(x^4)]]).
                        theta(y(x),x,2)+
      Matrix([[-1+2*x+O(x^2),5*x^2+x+O(x^4)],
              [O(x^2),11*x^2+O(x^4)]]).
                        theta(y(x),x,1)+
      Matrix([[O(1),-3*x^2+x+O(x^4)],
              [1+O(x^2),-6*x^2+O(x^4)]]).
                        y(x) = 0:
```

Получаем лорановы решения:

```
> s:=TruncatedSeries:-LaurentSolution(S1,y(x));
```

$$s := \left[\left[6\_c_1 x^2 + O(x^3), \frac{\_c_1}{x} + \_c_1 + O(x)\right]\right].$$

Применим процедуру Substitute к найденному решению:

```
> TruncatedSeries:-Substitute(s,S1,y(x));
```

$$\left[\left[O(x^2), O(x^3)\right]\right].$$

Изменим компоненты найденного усеченного решения

```
> s1:=[7*_c[1]*x^2+O(x^3),_c[1]/x+O(x)];
```

$$s1 := \left[7\_c_1 x^2 + O(x^3), \frac{\_c_1}{x} + O(x)\right]$$

и подставим в ту же систему:

```
> TruncatedSeries:-Substitute(s1,S1,y(x));
```

$$\left[-x\_c_1 + O(x^2), 7x^2\_c_1 + O(x^3)\right]$$

Видим, что усеченные ряды результата подстановки имеют ненулевые члены, что означает, что этот вектор не является решением системы.

Зададим систему (9):

```
> S2:=Matrix([[O(x^5),-1+O(x^5)],
              [1+O(x^5),O(x^5)]]).
                        theta(y(x),x,1)+
      Matrix([[O(x^5),O(1)],
              [2+O(x^5),O(x^5)]]).y(x)=0:
```

Ответ FAIL означает, что система не имеет лорановых решений:

```
> TruncatedSeries:-LaurentSolution(S2,y(x));
```

FAIL

Продолжения системы S2, которые имеют решения с разными началами:

```
> TruncatedSeries:-LaurentSolution(
    eval(S2,O(1)=5+O(x)),y(x));
```

$$\left[\left[O(x^{10}), \_c_1 x^5 + O(x^6)\right]\right]$$

```
> TruncatedSeries:-LaurentSolution(
    eval(S2,O(1)=6+O(x)),y(x));
```

$$\left[\left[O(x^{11}), \_c_1 x^6 + O(x^7)\right]\right]$$

БЛАГОДАРНОСТИ

Авторы благодарны С.А. Абрамову за полезные обсуждения рассматриваемой задачи и представленных результатов.

СПИСОК ЛИТЕРАТУРЫ

1. *Abramov S.A., Barkatou M.A., Khmelnov D.E.* On full rank differential systems with power series coefficients // J. Symbolic Comput. 2015. V. 68. P. 120–137.
2. *Abramov S.A.* EG-eliminations // J. Difference Equations Appl. 1999. V. 5. P. 393–433.
3. *Абрамов С.А., Рябенко А.А., Хмельнов Д.Е.* Поиск лорановых решений систем линейных дифференциальных уравнений с усеченными степенными рядами в роли коэффициентов // Программирование. 2023. № 5. С. 35–46.
4. *Абрамов С.А., Рябенко А.А., Хмельнов Д.Е.* Линейные обыкновенные дифференциальные уравнения и усеченные ряды // Ж. вычисл. матем. и матем. физ. 2019. Т. 59. № 10. С. 1706–1717.
5. *Khmelnov D.E., Ryabenko A.A.* Algorithm EG as a tool for finding Laurent solutions of linear differential systems with truncated series coefficients // Компьютерная

алгебра: материалы 5-й международной конференции. Москва, 26–28 июня 2023 г./ отв. ред. С.А. Абрамов, А.Б. Батхин, Л.А. Севастьянов. Москва: ИПМ им. М.В. Келдыша, 2023. С. 92–96.
6. Maple online help. http://www.maplesoft.com/support/help/
7. *Абрамов С.А., Рябенко А.А., Хмельнов Д.Е.* Процедуры поиска усеченных решений линейных дифференциальных уравнений с бесконечными и усеченными степенными рядами в роли коэффициентов // Программирование. 2021. № 2. С. 56–65.
8. *Абрамов С.А., Рябенко А.А., Хмельнов Д.Е.* Процедуры поиска усеченных решений линейных дифференциальных уравнений с бесконечными и усеченными степенными рядами в роли коэффициентов // Программирование. 2021. № 2. С. 56–65.

SEARCHING FOR LAURENT SOLUTIONS OF TRUNCATED SYSTEMS OF LINEAR DIFFERENTIAL EQUATIONS WITH THE USE OF EG-ELIMINATIONS

© 2024 A. A. Ryabenko[a], D. E. Khmelnov[a]

[a]Federal Research Center «Computer Science and Control» of Russian Academy of Sciences, ul. Vavilova 40, Moscow, 119333 Russia

Laurent solutions of systems of linear ordinary differential equations with the truncated power series coefficients are considered. The Laurent series in the solutions are also truncated. We use induced recurrent systems for constructing the solutions and have previously proposed an algorithm for the case when the induced system has a non-singular leading matrix. The algorithm finds the maximum possible number of terms of the series in the solutions that are invariant with respect to any prolongation of the original system. Below we present advances in extending our algorithm to the case when the leading matrix is singular using algorithm EG as an auxiliary tool. The implementation of the algorithm as a Maple procedure and examples of its usage are presented.

Keywords: truncated systems of linear ordinary differential equations, truncated Laurent solutions, computer algebra system Maple, EG-eliminations

ПРОГРАММИРОВАНИЕ, *2024*, *№ 2*, *с. 118–124*

КОМПЬЮТЕРНАЯ АЛГЕБРА

УДК 519.6

РЕШЕНИЕ ЗАДАЧ АНАЛИЗА РАЙСОВСКИХ ДАННЫХ: ТЕОРИЯ И ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ МЕТОДАМИ КОМПЬЮТЕРНОЙ АЛГЕБРЫ В СИСТЕМЕ WOLFRAM MATHEMATICA

© 2024 г. Т. В. Яковлева[a], * (ORCID: 0000-0003-2401-9825)

[a]Федеральный исследовательский центр "Информатика и управление" РАН

119333 Москва, ул. Вавилова, д. 44, к. 2, Россия

**E-mail: tan-ya@bk.ru*

Поступила в редакцию 01.07.2023
После доработки 16.08.2023
Принята к публикации 25.09.2023

В работе рассматриваются теоретические основы и математические методы анализа данных в условиях статистического распределения Райса. Поставленная задача предполагает совместный расчет параметров сигнала и шума. Показано, что такой расчет приводит к необходимости решения сложной системы существенно нелинейных уравнений с двумя неизвестными, что требует значительных вычислительных ресурсов. Представленное исследование направлено на математическую оптимизацию применения методов компьютерной алгебры для численного решения рассматриваемой задачи. В результате проведенной оптимизации решение системы двух нелинейных уравнений сводится к решению одного уравнения с одной неизвестной величиной, что существенно упрощает алгоритмы численного решения задачи, снижает объем необходимых вычислительных ресурсов и открывает перспективы использования развитых методов оценивания параметров в информационных системах с приоритетом работы в режиме реального времени. Результаты численных экспериментов, полученные с помощью использования системы Wolfram Mathematica, подтверждают эффективность разработанных методов двухпараметрического анализа райсовских данных. Рассматриваемые методы анализа данных являются значимыми для решения широкого круга научных и прикладных задач, в которых анализируемые данные описываются статистической моделью Райса.

Ключевые слова: статистическое распределение Райса, система нелинейных уравнений, система компьютерной алгебры Wolfram Mathematica, методы математической статистики

DOI: 10.31857/S0132347424020155 **EDN:** RNLIBR

1. ВВЕДЕНИЕ

При обработке стохастических данных, в частности, при решении задач шумоподавления, широко используется подход, основанный на использовании методов математической статистики, таких, в частности, как метод максимума правдоподобия, варианты метода моментов и т.д. Очевидно, что при применении данных методов эффективность решения задачи в значительной степени определяется особенностями статистического распределения анализируемых данных. В настоящей работе рассматривается проблема оптимизации применения методов компьютерной алгебры для решения задачи двухпараметрического анализа райсовских данных, который предполагает совместный расчет информативной и шумовой составляющих измеряемого сигнала.

Как известно, статистическое распределение Райса описывает широкий круг задач обработки информации, в которых выходной сигнал формируется как сумма искомого начального сигнала и случайного шума, генерируемого многими нормально-распределенными компонентами с нулевым средним значением и некоторой величиной стандартного отклонения. При этом измеряемой и анализируемой величиной является амплитуда, или огибающая результирующего сигнала, величина которой, как известно, подчиняется статистическому распределению Райса [1].

Задача совместного оценивания сигнала и шума при двухпараметрическом анализе райсовских данных состоит в совместном расчете параметров распределения Райса. Традиционно используемый однопараметрический подход к решению задачи восстановления полезного сигнала на фоне шума предполагает априорную известность одного из райсовских параметров, а именно – дисперсии шума, и, следовательно, расчет лишь одного неизвестного параметра [2–4]. Однако условие априорной известности дисперсии шума никогда не выполняется на практике. В противоположность упрощенному

однопараметрическому приближению, двухпараметрический подход не ограничен никакими априорными предположениями и тем самым обеспечивает значительно более корректное оценивание искомых величин сигнала и шума.

Однако решение задачи двухпараметрического анализа райсовских данных, как можно показать, связано с необходимостью решения системы двух существенно нелинейных уравнений с двумя неизвестными, что сопряжено со значительными трудностями как теоретического, так и вычислительного характера. В общем случае соотношения параметров такая задача может быть решена только численно, в связи с чем возникает проблема, обусловленная большим объемом необходимых вычислительных ресурсов. С другой стороны, задача расчета райсовских параметров, с решением которой связано восстановление полезного, информативного сигнала на фоне шума, особенно востребована во многих приложениях, которые предполагают необходимость обработки сигналов в системах с приоритетом работы в режиме реального времени. К таким приложениям, в частности, относится обработка данных в системах медицинской диагностики, в частности, анализ изображений при ультразвуковой визуализации.

В этой связи важно оптимизировать математические методы решения задачи двухпараметрического анализа райсовских данных таким образом, чтобы обеспечить упрощение алгоритмов численного решения задачи и сократить объем необходимых вычислительных ресурсов. Именно эти аспекты численного решения задачи анализа стохастических данных стали предметом данной работы: а именно, в результате проведенного математического исследования численное решение сложной системы двух нелинейных уравнений для двух неизвестных удалось свести к решению одного уравнения с одной неизвестной, что позволило обеспечить решение двухпараметрической задачи, используя компьютерные ресурсы лишь в том объеме, который необходим для решения однопараметрической задачи. Предварительная версия данной работы докладывалась на 5-й Международной конференции «Компьютерная алгебра» (Москва, июнь 2023 г.).

2. ТЕОРЕТИЧЕСКИЕ И ЧИСЛЕННЫЕ АСПЕКТЫ РЕШЕНИЯ ЗАДАЧИ СОВМЕСТНОГО РАСЧЕТА РАЙСОВСКИХ ПАРАМЕТРОВ

Как известно, статистическому распределению Райса подчиняется амплитуда случайной величины, образуемой в результате сложения исходно детерминированного комплексного сигнала и искажающего его гауссовского шума. В силу центральной предельной теоремы такая ситуация является весьма типичной для многих задач обработки сигналов различной физической природы. Это означает, что статистическая модель Райса описывает широкий спектр задач, что обусловливает значимость разработки и широкую применимость методов анализа райсовских данных с целью восстановления исходного, неискаженного сигнала на фоне шума.

В задачах анализа райсовского сигнала измеряемой величиной является амплитуда

$$x = \sqrt{x_{\mathrm{Re}}^2 + x_{\mathrm{Im}}^2}$$

комплексной переменной величины с действительной x_{Re} и мнимой x_{Im} составляющими, характеризуемыми математическим ожиданием ν и искажаемыми нормально распределенным гауссовским шумом с дисперсией σ^2 [1–5]. Амплитуда

$$x = \sqrt{x_{\mathrm{Re}}^2 + x_{\mathrm{Im}}^2}$$

подчиняется статистическому распределению Райса с функцией плотности вероятности, определяемой следующим выражением:

$$P\left(x\middle|\nu,\sigma^2\right) = \frac{x}{\sigma^2}\exp\left(-\frac{x^2+\nu^2}{2\sigma^2}\right)\cdot I_0\left(\frac{x\nu}{\sigma^2}\right), \quad (1)$$

где $I_\alpha(z)$ — модифицированная функция Бесселя первого рода порядка α. Поставленная задача двухпараметрического анализа райсовских данных состоит в совместном расчете параметров ν и σ^2 распределения Райса на основе данных, полученных в выборках измерений. Именно эти параметры определяют величину исходного, неискаженного сигнала ν и дисперсию искажающего его шума σ^2.

На рис.1 представлена иллюстрация формирования райсовского сигнала из исходно детерминированной величины $A=\nu$ амплитуды вектора, характеризующего некоторый процесс и искажаемого под неизбежным воздействием гауссовского шума, описываемого неким вектором $\vec{r}$.

На рис. 1 амплитуда x является райсовской величиной, поведение функции плотности вероятности которой иллюстрируется таким образом, что яркость каждой точки на рис. 1 пропорциональна значению данной функции в соответствующей точке.

Величины математического ожидания $\bar{x}$ и квадрата стандартного отклонения σ_x^2 даются следующими формулами:

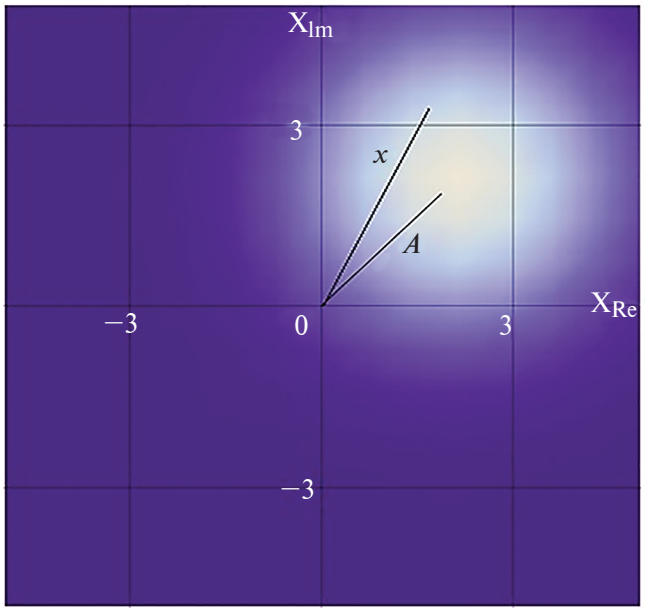


Рис. 1. Иллюстрация поведения функции плотности вероятности, характеризующей распределение райсовской величины x, формируемой из исходной детерминированной величины A под воздействием гауссовского шума.

$$\overline{x} = \sigma\sqrt{\pi/2}\cdot L_{1/2}(-\nu^2/2\sigma^2),$$

$$\sigma_x^2 = 2\sigma^2 + \nu^2 - \frac{\pi\sigma^2}{2}\cdot L_{1/2}(-\nu^2/2\sigma^2), \qquad (2)$$

где $L_{1/2}(z)$ — полином Лаггера. Как легко видеть, математическое ожидание райсовской величины и ее стандартное отклонение не совпадают с параметрами ν и σ^2, что характеризует особые, нелинейные свойства этого статистического распределения, в силу которых анализ райсовской случайной величины требует развития особых методов и соответствующего математического аппарата.

Ряд математических методов совместного расчета райсовских параметров сигнала и шума были развиты и строго обоснованы в работе [5] и других работах автора данной статьи. В основе этих методов лежат принципы математической статистики. В данной работе возможность существенной оптимизации решения задачи методами компьютерной алгебры продемонстрирована на примере использования для расчета райсовских параметров метода, представляющего собой комбинирование принципа максимума правдоподобия и метода моментов.

Ниже приведены основные выкладки, лежащие в основе данного метода. Будем использовать следующие обозначения: x_i — величина сигнала, полученная как результат i-го измерения в выборке; n — количество элементов в выборке, называемое также

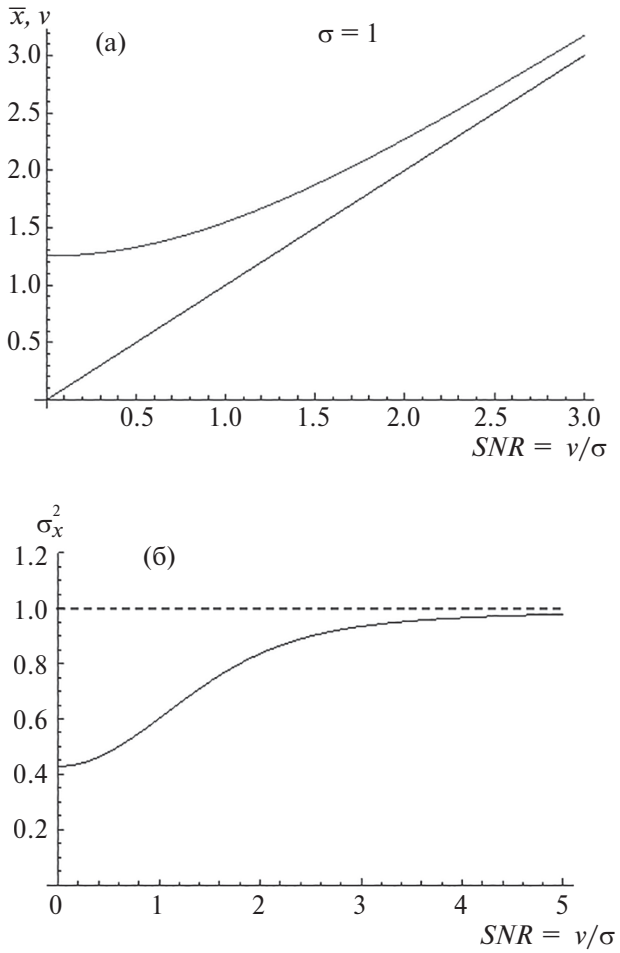


Рис. 2. Иллюстрация нелинейных свойств распределения Райса: (а) – нелинейная зависимость математического ожидания райсовской величины $\overline{x}$ от райсовского параметра ν; (б) – нелинейная зависимость квадрата стандартного отклонения райсовского сигнала от дисперсии гауссовского шума σ^2, формирующего райсовскую случайную величину.

длиной выборки. Рассчитанное на основе выборочных измерений $x_i\,(i = 1,\ldots,n)$ значение $\langle x^k\rangle$ стремится к значению соответствующего k-го момента $\overline{x^k}$ случайного райсовского сигнала при бесконечно большой длине n выборки измерений:

$$\overline{x^k} = \lim\frac{1}{n}\sum_{i=1}^{n} x_i^k.$$

Рассмотрим выборку из n измерений величины райсовской случайной величины x. Тогда функция правдоподобия $F(\nu,\sigma^2)$, т.е. функция совместной плотности вероятности событий, состоящих в том, что результатом i-го измерения является величина $x_i\,(i = 1,\ldots,n)$, выражается как произведение функций плотности вероятности для каждого измерения из выборки:

$$F(\nu,\sigma^2)=\prod_{i=1}^{n}P(x_i|\nu,\sigma^2), \qquad (3)$$

где функция $P(x|\nu,\sigma^2)$ определяется формулой (1). При полученных в результате измерений конкретных данных выборок функция правдоподобия является функцией неизвестных статистических параметров ν и σ^2. Как известно, метод максимума правдоподобия состоит в определении тех значений параметров ν и σ^2, которые максимизируют функцию правдоподобия (3), или, что эквивалентно, ее логарифм, называемый логарифмической функцией правдоподобия распределения Райса:

$$\ln F\left(\nu,\sigma^2\right)=$$

$$\sum_{i=1}^{n}\left\{\ln x_i-\ln\sigma^2-\frac{x_i^2+\nu^2}{\sigma^2}+\ln I_0\left(\frac{x_i\nu}{\sigma^2}\right)\right\}. \qquad (4)$$

Существование и единственность максимума функции правдоподобия были строго доказаны автором данной работы и иллюстрируются рис. 3, который представляет трехмерные графики функции правдоподобия распределения Райса, полученные в ходе численного эксперимента в системе Wolfram Mathematica для различных значений отношения параметров сигнала и шума. Ожидаемо, чем больше значение величины отношения сигнала к шуму, тем более детерминированным становится сигнал и тем более острую форму приобретает функция правдоподобия.

Представляемый в настоящей работе метод определения искомых параметров ν и σ^2 распределения Райса основан на использовании двух известных подходов к решению задач математической статистики: принципа максимума правдоподобия и метода моментов [6–8]. В качестве одного из уравнений нового комбинированного метода будем использовать уравнение метода максимума правдоподобия, полученное приравниванием нулю частной производной логарифмической функции правдоподобия по параметру сигнала ν:

$$\frac{\partial}{\partial\nu}\ln F(\nu,\sigma^2)=0.$$

Тогда, принимая во внимание выражение (4), получаем:

$$\sum_{i=1}^{n}\frac{\partial}{\partial\nu}\ln I_0\left(\frac{x_i\nu}{\sigma^2}\right)-\frac{n\cdot\nu}{\sigma^2}=0. \qquad (5)$$

В качестве второго уравнения комбинированного метода будем использовать выражение для первого момента райсовской случайной величины [9]:

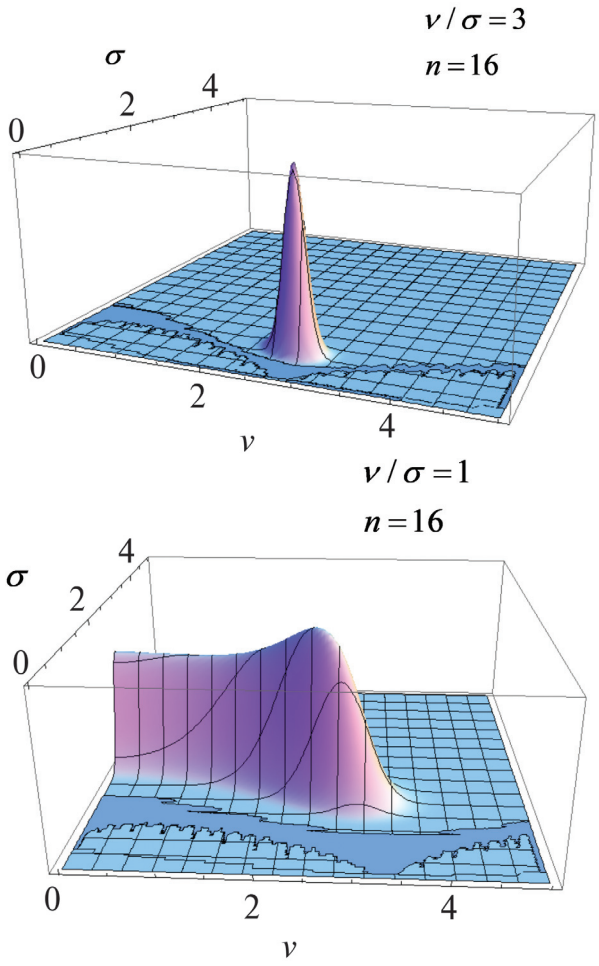


Рис. 3. Трехмерные графики функции правдоподобия статистического распределения Райса, построенные в системе Wolfram Mathematica для различных соотношений величин райсовских параметров.

$$\bar{x}=\sigma\sqrt{\pi/2}\cdot L_{1/2}(-\nu^2/2\sigma^2). \qquad (6)$$

Формулы (5) и (6) фактически представляют собой систему двух уравнений для двух неизвестных ν и σ^2.

Используя известное соотношение

$$\frac{d}{dz}I_0(z)=I_1(z),$$

[10] и проводя несложные преобразования, получаем следующую систему уравнений рассматриваемого метода:

$$\begin{cases}\nu=\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}x_i\cdot\tilde{I}\left(\frac{x_i\nu}{\sigma^2}\right),\\ \dfrac{1}{n}\displaystyle\sum_{i=1}^{n}x_i=\sigma\sqrt{\frac{\pi}{2}}\cdot L_{1/2}\left(-\frac{\nu^2}{2\sigma^2}\right).\end{cases} \qquad (7)$$

В первом уравнении системы (7) обозначение $\tilde{I}$ используется для функции

$$\tilde{I}(z) = \frac{I_0(z)}{I_1(z)},$$

равной отношению модифицированных функций Бесселя первого и нулевого порядков. Уравнения (7) представляют собой систему двух существенно нелинейных уравнений для двух неизвестных: ν и σ^2.

Введем переменную $r = \nu^2/(2\sigma^2)$, которая характеризует величину отношения сигнала к шуму. Переходя от пары переменных (ν,σ) к переменным (r,σ), получим из (7) следующую систему уравнений:

$$\begin{cases} \sigma\sqrt{2r} = \dfrac{1}{n}\sum\limits_{i=1}^{n} x_i \cdot \tilde{I}\left(\dfrac{x_i}{\sigma}\sqrt{2r}\right), \\ \langle x \rangle = \sigma\sqrt{\pi/2}\cdot L_{1/2}(-\nu^2/2\sigma^2), \end{cases} \quad (8)$$

где $\langle x \rangle = \dfrac{1}{n}\sum\limits_{i=1}^{n} x_i$ — измеренное в выборках значение первого момента, или среднего значения райсовской величины x. Из второго уравнения системы (8) получаем для переменной σ следующее выражение:

$$\sigma = \frac{\langle x \rangle}{\sqrt{\dfrac{\pi}{2}}\cdot L_{1/2}(-r)}. \quad (9)$$

Подставляя (9) в первое уравнение системы (8), получаем уравнение для переменной r:

$$2\sqrt{\frac{r}{\pi}}\frac{\langle x \rangle}{L_{1/2}(-r)} = \frac{1}{n}\sum_{i=1}^{n} x_i \tilde{I}\left(\frac{x_i\sqrt{\pi r}}{\langle x \rangle}L_{1/2}(-r)\right). \quad (10)$$

Таким образом, задачу решения системы (8) двух уравнений для двух неизвестных (r,σ) удалось свести к задаче решения одного уравнения (10) для одной неизвестной величины r, что позволяет существенно упростить решения данной задачи посредством символьных вычислений в системе Wolfram Mathematica.

Поставленная задача расчета величины полезного сигнала и дисперсии шума состоит в том, чтобы, подставляя выборочные данные для величины сигнала $x_i\,(i = 1,\ldots,n)$ в уравнение (10), найти решение этого уравнения для неизвестной величины r и затем рассчитать параметр шума σ, используя формулу (9). Используя полученные значения для σ и r, а также формулу $r = \nu^2/(2\sigma^2)$, нетрудно получить значение искомого параметра полезного сигнала $\nu = \sqrt{2\sigma^2 r}$.

На рис. 4 представлены графики расчета райсовских параметров, полученные представленным выше

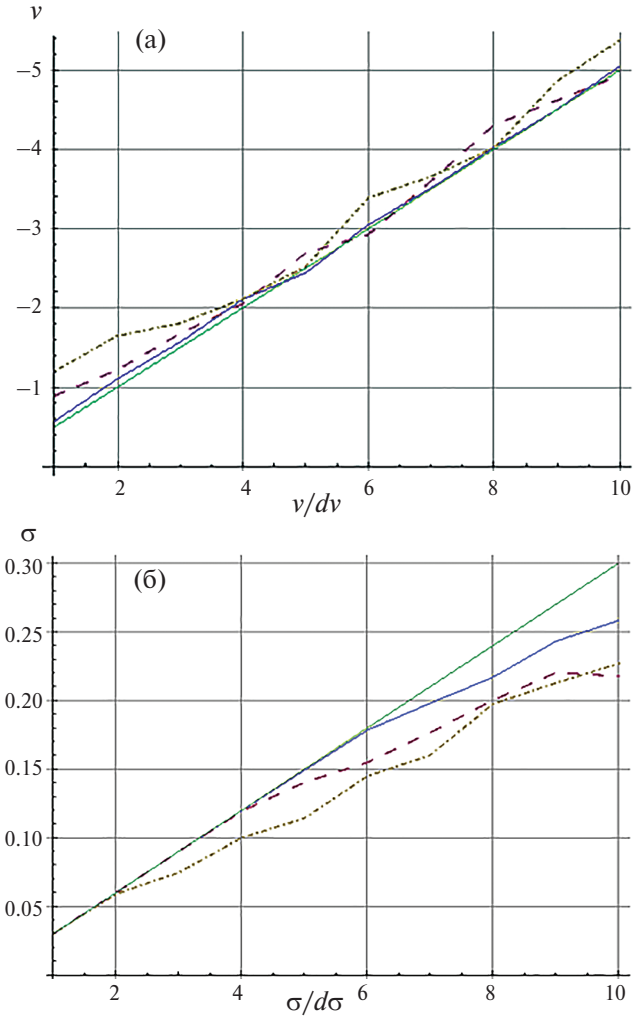


Рис. 4. Результаты расчета райсовских параметров сигнала ν (а) и шума σ (б) в системе Wolfram Mathematica посредством представленного комбинированного метода.

методом в ходе символьных вычислений в системе Wolfram Mathematica.

На рис. 4 а представлена зависимость рассчитанного параметра сигнала от величины, пропорциональной отношению сигнала к шуму, при заданном значении второго параметра, причем сплошная, пунктирная и штрихпунктирная кривые соответствуют следующим значениям параметра: =0.5 (сплошная линия); (пунктирная линия); =1,5 (штрихпунктирная линия). На рис. 4 б представлены зависимости рассчитанного параметра шума при различных заданных значениях параметра сигнала, а именно: =3.0, =2.0 и =1.0, соответственно. Прямые линии на обоих графиках соответствуют исходно заданным значениям параметров. Длина выборки для проводимых вычислений составляла с усреднением по 25 выборкам (в реальных системах обра-

ботки райсовских данных число усредняемых выборок составляет, как правило, величины порядка $10^3 \div 10^4$, что обеспечивает гораздо более высокую точность расчетов. Представленные графики демонстрируют также ожидаемое повышение точности расчетов с увеличением величины отношения сигнала к шуму.

3. ЗАКЛЮЧЕНИЕ

Совместный расчет параметров статистического распределения Райса обеспечивает эффективное восстановление информативной составляющей стохастических данных на фоне шума. Представленное исследование дает возможность существенно сократить вычислительные ресурсы, требуемые для решения данной задачи, и обеспечивает упрощение алгоритмов численного анализа райсовских данных и тем самым – оптимизацию методов компьютерной алгебры для совместного расчета райсовских параметров сигнала и шума. В результате задача решения системы двух существенно нелинейных уравнений для двух неизвестных была сведена к решению одного уравнения для одной неизвестной величины, что означает существенное повышение эффективности методов компьютерной алгебры для решения рассматриваемой задачи, а также значительное, практически двукратное сокращение требуемых вычислительных ресурсов: представленные методы обеспечивают решение двухпараметрической задачи, используя вычислительные ресурсы лишь в объеме, необходимом для решения однопараметрической задачи.

Такая оптимизация решения задачи методами компьютерной алгебры и снижение объема необходимых вычислительных ресурсов позволяют в свою очередь повысить информативность и точность результатов обработки стохастических данных, что, в свою очередь, делает возможным применение разработанных методов в информационных технологиях и системах с приоритетом работы в режиме реального времени. Эта возможность очень важна во многих приложениях, в частности, таких, которые связаны с обработкой изображений в системах медицинской диагностической визуализации.

Численные результаты подтверждают эффективность решения задачи анализа райсовских данных разработанными методами с обеспечением высокой точности в широком диапазоне величин соотношения параметров.

Представленная математическая оптимизация продемонстрировала, что совершенствование средств компьютерной алгебры с целью упрощения символьных вычислений является важной и решаемой задачей.

БЛАГОДАРНОСТИ

Автор выражает глубокую признательность Абрамову Сергею Александровичу и Рябенко Анне Андреевне за полезные обсуждения и помощь в подготовке данной работы.

СПИСОК ЛИТЕРАТУРЫ

1. *Rice S. O.* Mathematical analysis of random noise // Bell Syst. Technological J. 1944. V. 23. P. 282.
2. *Benedict T.R., Soong T.T.* The joint estimation of signal and noise from the sum envelope IEEE Transactions on Information Theory. Institute of Electrical and Electronics Engineers. 1967. V. 13. № 3. P. 447–454.
3. *Talukdar K.K., Lawing W.D.* Estimation of the parameters of Rice distribution ,J. Acoust. Soc. Amer., Mar. 1991. V. 89. № 3. P. 1193–1197.
4. *Sijbers J., den Dekker A.J., Scheunders P., Van Dyck D.* Maximum-Likelihood Estimation of Rician Distribution Parameters, IEEE Transactions on Medical Imaging. 1998. V. 17. № 3. P. 357–361.
5. *Yakovleva T.V.* A Theory of Signal Processing at the Rice Distribution, Dorodnicyn Computing Centre, RAS, Moscow, 2015, 268 p.
6. *Deutsch R.* Estimation Theory. NJ: Prentice-Hall: Englewood Cliifs, 1965.
7. *Port S.C.* Theoretical Probability for Applications. New York: Wiley, 1944.
8. *Venttsel' E.S.,* Teoriya veroyatnostei (Probability Theory), Moscow: Akademiya, 2005, 10th ed.
9. *Park J.H.* Moments of the generalized Rayleigh distribution // Quarterly of Applied Mathematics. 1961. V. 19. № 1. P. 45–49.
10. *Abramowitz, M., Stegun, I.A.* Handbook of Mathematical Functions, United States Department of Commerce, National Bureau of Standards (NBS), 1964.

SOLVING RICIAN DATA ANALYSIS PROBLEMS: THEORY AND NUMERICAL MODELING USING COMPUTER ALGEBRA METODS IN WOLFRAM MATHEMATICA

© 2024 T. V. Yakovleva[a]

[a] *Federal Research Center "Computer Science and Control", Russian Academy of Sciences, ul. Vavilova 44/2, Moscow, 119333 Russia*

This paper considers theoretical foundations and mathematical methods of data analysis under the conditions of the Rice statistical distribution. The problem involves joint estimation of the signal and noise parameters. It is shown that this estimation requires the solution of a complex system of essentially nonlinear equations with two unknown variables, which implies significant computational costs. This study is aimed at mathematical optimization of computer algebra methods for numerical solution of the problem of Rician data analysis. As a result of the optimization, the solution of the system of two nonlinear equations is reduced to the solution of one equation with one unknown variable, which significantly simplifies algorithms for the numerical solution of the problem, reduces the amount of necessary computational resources, and enables the use of advanced methods for parameter estimation in information systems with priority of real-time operation. Results of numerical experiments carried out using Wolfram Mathematica confirm the effectiveness of the developed methods for two-parameter analysis of Rician data. The data analysis methods considered in this paper are useful for solving many scientific and applied problems that involve analysis of data described by the Rice statistical model.

Keywords: statistical Rice distribution, system of nonlinear equations, computer algebra system Wolfram Mathematica, methods of mathematical statistics