SCSS 2024 WiP

Pre-Proceedings of the

10th International Symposium on Symbolic Computation in Software Science *Works in Progress Workshop*

Tokyo, Japan, August 28-30 2024

Edited by
Katsusuke Nabeshima *Tokyo University of Science*
Stephen M. Watt *University of Waterloo*

Preface

SCSS 2024 is the 10th edition of the *10th International Symposium on Symbolic Computation in Software Science*. The symposium aims to promote research on the theoretical and practical aspects of symbolic computation in software science in the context of modern computational and artificial intelligence techniques. It will be held in Tokyo from August 28 to 30.

The symposium has three main types of presentations:

- the keynote and invited talks
- formal full papers
- works in progress.

This volume contains the record of the Work in Progress of SCSS 2024. The formal full papers and the abstracts of the keynote and invited talks appear in the Springer Lecture Notes series as LNAI 14991.

What is the meaning of the symposium name "symbolic computation in software science"? Symbolic computation is the science of computing with symbolic objects (terms, formulae, programs, representations of algebraic objects, and so on). Powerful algorithms have been developed during the past decades for the significant subareas of symbolic computation: computer algebra and computational logic. These include resolution proving, model checking, provers for various inductive domains, rewriting techniques, cylindrical algebraic decomposition, Gröbner bases, characteristic sets, and telescoping for recurrence relations. These algorithms and methods have been successfully applied in various fields. Software science has the goal of applying scientific principles in the development of software and covers a broad range of topics in software construction and analysis. One of the main objectives is to enhance software quality. The SCSS meetings bring these fields together, allowing the ideas from each to enhance the other.

Over the years, the scope of SCSS has evolved, incorporating new research themes that drive progress in symbolic computation in software science. Some of the recurring topics in the SCSS meetings have been:

- Theorem proving methods and techniques
- Algorithm synthesis and verification
- Formal methods, including for the analysis of network security
- Complexity analysis and termination analysis of algorithms
- Extraction of specifications from algorithms
- Generation of inductive assertions for algorithms
- Algorithm transformations
- Component-based programming
- Symbolic methods for semantic web and cloud computing.

The present instance of SCSS builds on these themes.

The abstracts and papers presented here emphasize symbolic computation, formal systems, and applications of formal methods. After fifteen years, the foundational framework stands firm, continually incorporating innovative developments in SCSS domains.

August 2024

Katsusuke Nabeshima
Stephen M. Watt

Organization

SCSS Steering Committee *as of August 2023*

Adel Bouhoula	Arabian Gulf University, Bahrain
Bruno Buchberger	RISC Johannes Kepler University, Austria
Hoon Hong	North Carolina State University, USA
Tetsuo Ida	University of Tsukuba, Japan
Laura Kovács	TU Wien, Austria
Temur Kutsia	RISC Johannes Kepler University, Austria
Mohamed Mosbah	LABRI, France
Michael Rusinowitch	INRIA, France
Masahiko Sato	Kyoto University, Japan
Carsten Schneider	RISC Johannes Kepler University, Austria
Dongming Wang	Beihang University, China, and CNRS, France

SCSS 2024 Organizing Committee

General Chair	Tetsuo Ida	U. Tsukuba
Program Committee Chair	Stephen Watt	U. Waterloo
Local Arrangements Chair	Katsusuke Nabeshima	Tokyo U. of Science

Program Committee

David Cerna	Czech Academy of Sciences, Czechia
Changbo Chen	Chinese Academy of Sciences, China
Rachid Echahed	CNRS and University of Grenoble, France
David Jeffrey	University of Western Ontario, Canada
Cezary Kaliszyk	University of Innsbruck, Austria
Yukiyoshi Kameyama	University of Tsukuba, Japan
Laura Kovács	TU Wien, Austria
Temur Kutsia	RISC Johannes Kepler University, Austria
Christopher Lynch	Clarkson University, USA
Yasuhiko Minamide	Tokyo Institute of Technology
Julien Narboux	CNRS and Université de Strasbourg, France
Wolfgang Schreiner	RISC Johannes Kepler University, Austria
Sofiène Tahar	Concordia University, Canada
Stephen Watt (chair)	University of Waterloo, Canada
Lihong Zhi	AMSS Chinese Academy of Sciences, China

Local Arrangements Committee

Yuki Ishihara	Nihon University, Japan
Katsusuke Nabeshima (chair)	Tokyo University of Science, Japan
Yosuke Sato	Tokyo University of Science, Japan
Hiroshi Sekigawa	Tokyo University of Science, Japan
Akira Terui	University of Tsukuba, Japan

Sponsors

SCSS 2024 gratefully acknowledges the support of our sponsors,
the Kayamori foundation of informational science advancement and Maplesoft.

Maplesoft™
Mathematics • Modeling • Simulation
A Cybernet Group Company

Table of Contents

Improving LLM-based Code Completion Using LR Parsing-Based Candidates

Md Monir Ahammod Bin Atique[1,†], Kwanghoon Choi[1,*,†], Isao Sasano[2] and Hyeon-Ah Moon[3]

[1]*Chonnam National University, Gwangju 61186, South Korea*
[2]*Shibaura Institute of Technology, Tokyo, Japan*
[3]*Sogang University, Seoul, South Korea*

Abstract

Programmers often use syntax completion and code suggestion features. Our methodology enhances code completion by combining structural candidate information from LR parsing with LLMs. These structural candidates are utilized to compose prompts so that ChatGPT can predict actual code under the specified structure. Tested on Small Basic and C benchmarks, this approach offers textual suggestions rather than just structural ones, showing nearly 50% prediction accuracy for Small Basic programs. While effective for Small Basic, we report that challenges remain with C11 programs.

Keywords

Syntax Completion, Large Language Model, LR parsing, Integrated Development Environments

1. Introduction

Many integrated development environments (IDEs), such as Visual Studio Code, provide syntax completion features that ease the editing process for various programming languages. Developers of IDEs should prioritize incorporating syntax completion for each supported language. To make the process more efficient and cost-effective, it is beneficial to approach this implementation methodically, guided by a detailed specification.

An analytic approach is based on syntax analysis using the well-developed LR parsing theory [1]. Sasano & Choi [2] defined code completion candidates γ for a prefix $\alpha\beta$ as suffix sentential forms derived from a start symbol S if there is a production $A \to \beta\gamma$ in the LR grammar so that $\beta\gamma$ can be reduced to a nonterminal A. Figure 1 describes this idea.

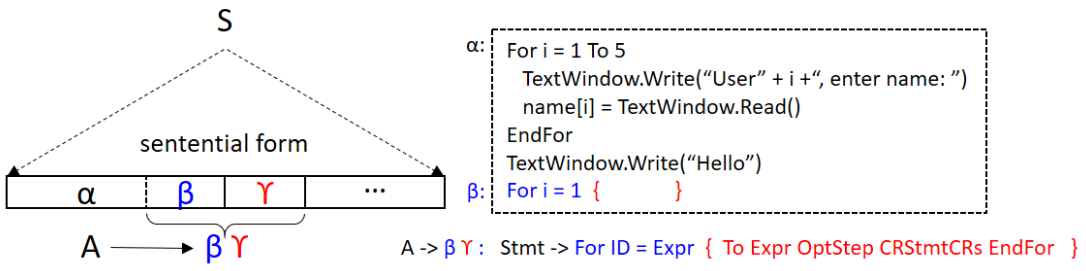

S
sentential form
α
β
ϒ
...
A → βϒ
α:
For i = 1 To 5
TextWindow.Write("User" + i +", enter name: ")
name[i] = TextWindow.Read()
EndFor
TextWindow.Write("Hello")
β:
For i = 1 { }
A -> βϒ : Stmt -> For ID = Expr { To Expr OptStep CRStmtCRs EndFor }

Figure 1: The idea of structural candidates for code completion using LR parsing [2]

For example, on a request for a code completion on (the part of) a prefix 'For i = 1', β is 'For ID = Expr', which is a sequence of terminal and noterminal symbols describing the beginning of the for loop.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
*Corresponding author.
†These authors contributed equally.
monir024@jnu.ac.kr (M. M. A. B. Atique); kwanghoon.choi@jnu.ac.kr (K. Choi); sasano@sic.shibaura-it.ac.jp (I. Sasano); hamoon@sogang.ac.kr (H. Moon)
https://monircse061.github.io/page/ (M. M. A. B. Atique); https:https://kwanghoon.github.io/ (K. Choi)
0009-0000-7103-9744 (M. M. A. B. Atique); 0000-0003-3519-3650 (K. Choi); 0000-0002-9373-6206 (I. Sasano); 0000-0001-7359-3298 (H. Moon)

Sasano & Choi's method [2] [3] can automatically uncover a candidate γ, which is 'To Expr OptStep CRStmtCRs EndFor' to complete the rest of the for loop by a production 'Stmt $\rightarrow$ For ID = Expr To Expr OptStep CRStmtCRs EndFor'. Consequently, IDEs will respond with this candidate γ to the request for a code completion on 'For i = 1'. Their continuing research [4] proposed a ranking method useful to choose more likely candidate than the others when there are more than one candidate possible for a prefix. It pre-investigates the frequencies of the occurrences of the candidates in the existing open-source projects.

These methods [2][3] [4] are advantageous. The suggested candidates are guaranteed to be correct syntactically, ranking can be customized for an individual software project, and this method can be implemented in a programming language agnostic way.

However, the suggested candidates by the methods are limited to the form of terminal and noterminal symbols. After choosing a candidate, programmers should manually edit it into a code text, which diminishes productivity. Determining such a code text for a candidate is beyond the LR parsing-based syntax analysis.

In this work, we study how Large Language Model [5] can complement these methods. Given a prefix text for $\alpha\beta$, the LR parsing based method firstly suggests a candidate γ, and the LLM produces a code completion satisfying the structure of the suggested candidate for the given prefix text. For example, our system can automatically compose a prompt to the LLM as

```
This is the incomplete Small Basic programming language code:
   1: For i = 1 To 5
   2: TextWindow.Write("User" + i + ", enter name: ")
   3: name[i] = TextWindow.Read()
   4: EndFor
   5: TextWindow.Write("Hello ")
   6: For i = 1 {To Expr OptStep CRStmtCRs EndFor}

Complete the {To Expr OptStep CRStmtCRs EndFor} part of the code.
Just show your answer in place of {To Expr OptStep CRStmtCRs EndFor}.
```

where the suggested structural candidate is placed inside the braces. Then the LLM successfuly returned exactly what we expected as this.

```
   6:           To 5
   7: TextWindow.Write(name[i] + ", ")
   8: EndFor
```

Thus the two approaches can complement each other. The LR parsing based analytic approach can precisely specify the syntactic code structure to complete, while the LLM-based statistical approach can predict the code text under the specified structure. According to [4], the top 1.8 suggested candidates in the SmallBasic programs and the top 3.15 suggested candidates in the C11 programs on average were found to be what are expected for testing. This evaluation results imply that candidates in the form of the rest structural candidates should not be considered by the LLM. Composing prompts using the top suggested structual candidates will be effective to instruct the LLM to exclude the bottom ones for code completion.

To the best of our knowledge, this is the first attempt to guide an LLM using prompts that utilize candidate structural information obtained from LR-parsing. We report ongoing work in this direction.

Our contributions are as follows.

Firstly, we propose a code completion prediction method that combines LR-parsing-based ranking of candidate skeletons with Large Language Model (LLM)-based fleshing out of those skeletons.

Secondly, we have setup an environment to evaluate the proposed method and report initial results using SmallBasic and C11 benchmarks.

Section 2 introduces our system and presents initial evaluation results . Section 3 compares our work with existing research. Section 4 concludes the paper with future work.

2. An Overview of Our System and Its Evaluation

Figure 2 shows an overview of our system. The system operates in two phases. The collection phase constructs a database from sample programs, mapping parse states to sets of ranked candidates. The query phase retrieves a sorted list of candidates based on their ranks for a parse state corresponding to a given cursor position being edited. The top suggested structural candidates are chosen from the sorted list to compose a prompt, and then the LLM fleshes out the structural candidates to produce textual candidates, which will be displayed to the programmer for code completion.

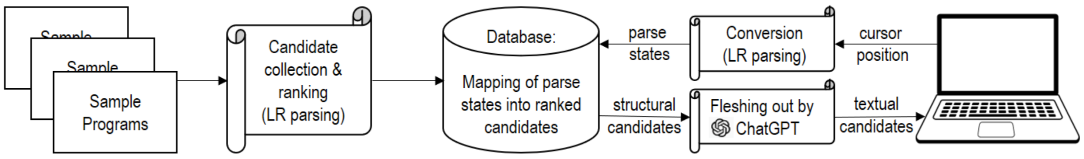


Figure 2: Overview of our system

In this work, we focus on one aspect of this system: automatically composing prompts to the LLM using structural candidates offered by the LR-based method is feasible and is an advancement to the previous work [4] in that this system can now suggest textual candidates rather than structural candiates.

Under this goal, we design an experiment to evaluate the effectiveness of using ChatGPT for code completion suggestions and to offer structural candidates (composed of terminals and nonterminals) to guide these suggestions. Our proposed system can be assessed by addressing the following two research questions (RQ):

- RQ1: Does the proposed system offer textual (actual) candidate suggestions with the aid of LLM such as ChatGPT that are beneficial in introductory programming?
- RQ2: Is it reasonable to implement the system as a language-parametric tool?

In order to address the above research questions, our methodology includes the following steps. **Selection of Programming Languages:** We selected two programming languages, Microsoft SmallBasic (MSB) and C11, for our experiments as in the previous work [4]. These languages are popular choices for introductory programming. **Data Collection:** The testing set for SmallBasic was obtained from its community. It consists of 27 programs totaling 155 lines taken from the well-known MSB tutorial. Talking about C, the test set of C11 comprises 106 programs (11,218 lines in total) that are solutions from the well-known book on the C programming language by Kernighan and Ritchie. **Prefix Extraction:** Prefixes were collected from the source code files, along with cursor position information. This information was obtained from candidates' database by the lexical analysis in the LR parsing-based method [4]. **Prompt Engineering:** Using the collected prefix and structural candidate data, we crafted prompts for ChatGPT. In this experiment, we selected the 'gpt-3.5-turbo-0125' model for its better performance with code completion. These information were then fed into the ChatGPT prompt to ask for substitutes for our structural candidates into actual candidates. We compared the answers provided by ChatGPT with the correct answers from our database. **Evaluation:** The responses from ChatGPT were evaluated using well-known techniques such as SacreBLEU, which is a popular method for assessing large language models and SequenceMatcher similarity. The SacreBLEU score measures the n-gram (sequences of n items, typically words or characters) similarity between the reference code sequence and the generated code sequence. It counts how many n-grams in the generated code sequence match (token-by-token) n-grams in the the reference code sequence. The SequenceMatcher is a class available in python module named "difflib". It compares the similarity between two sequences of strings (in terms of characters) by identifying the best alignment between them. Given two sequences, find the length of the longest subsequence present in both of them. Here, we used the parameter isjunk=None so that no elements are ignored. The data set as well as the developed software are all available in the public repository [1].

[1]https://github.com/monircse061/ChatGPT-Code-Completion-Work

We present a summary of the experimental results for both MSB and C11 which is dipicted in Table 1. For MSB, we experimented with 27 programs where, for each program, we iterated our system for each structural candidate, calculated the evaluating metrics values, and then averaged the precision for the whole program. This process was done for every program. Finally, we calculated the mean precision for the 27 programs in terms of SacreBLEU and sequence matcher similarity. On average, our system predicts the textual code suggestion with over 45% accuracy for each testing program when using SacreBLEU as an evaluation metric. Precision is almost similar at nearly 45% when sequence matcher similarity is taken into account. The similar process was used with C11. For 106 C11 programs, the average SacreBLEU score is 21.463%, indicating that our system forecasts the correct code completion suggestions. Sequence matcher similarity is nearly the same for C11.

Table 1
Experimental results (precision) on specific languages

PLs	Microsoft SmallBasic (%)	C11 (%)
SacreBLEU Precision	45.247	21.463
Sequence Matcher Precision	44.354	20.384

To show the effectiveness of guidance by a structural candidate, we discuss a case representing the best prediction of our system as this. In the MSB experiment case depicted in Figure 3, line 2600 marks the parse state and cursor position, followed by the next few lines (2602 to 2612), which provide the prompt for the ChatGPT. Lines spanning from 2603 to 2608 represent the prefix code. Subsequently, a candidate structure appears in line 2609: ‘To Expression OptStep CRStmtCRs EndFor’. It interprets that the actual candidate should be ‘To 5 \n TextWindow . Write (name [i] + ", ") \n EndFor’, which is shown in line 2618. Line 2614 outlines the time taken from the query to the ChatGPT response, which is 0.6903 seconds. In this candidate structure, the response generated by ChatGPT is highly accurate. The precision at the unigram level (1-gram) is 100% which is seen at line 2621, and other metric also show satisfactory result (line 2622). This example demonstrates that our candidate suggestion plays a crucial role in guiding ChatGPT’s responses. Each terminal and non-terminal component contributes to achieving an accurate result from ChatGPT.

```
2600:  Parse State: 85Cursor Position: 6 11
2601:
2602:  This is the incomplete Small Basic programming language code:
2603:  For i = 1 To 5
2604:       TextWindow.Write("User" + i + ", enter name: ")
2605:       name[i] = i
2606:  EndFor
2607:  TextWindow.Write("Hello ")
2608:  For i = 1
2609:                 ‘To Expression OptStep CRStmtCRs EndFor’
2610:  Complete the ‘To Expression OptStep CRStmtCRs EndFor’ part of the code
2611:  in the Small Basic programming language. Just show your answer in place
2612:  of ‘To Expression OptStep CRStmtCRs EndFor’.
2613:
2614:  Time taken: 0.6902937889099121 seconds
2615:  Received response: To 5
2616:      TextWindow.Write(name[i] + ", ")
2617:  EndFor
2618:  Actual result:  To 5 \n TextWindow . Write ( name [ i ] + ", " ) \n EndFor
2619:  SACREBLEU Score: {‘precisions’: [100.0, 86.66666666666667, 78.57142857142857,
2620:  76.92307692307692], ‘sys_len’: 16, ‘ref_len’: 20}
2621:  First element of precision:100.0
2622:  Sequence Matcher Similarity Precision:0.8462619469026548
```

Figure 3: High Prediction Result (MSB)

Based on the evidence provided, we can answer Research Question 1 in the affirmative.

Using ChatGPT with LR parsing-based structural candidates is effective in providing code completion suggestions for introductory programming languages, particularly for MSB. Our system shows correct suggestions with minimal prefixes (hints), which is notable. This indicates that the system can be beneficial in educational contexts where MSB is used. However, improvements are needed to increase precision, especially for more complex languages like C11. The precision for C programs in C11 is low due to short candidate structures like '[', ';' and complex, hard-to-infer structures. Additionally, predicting the next token or line of code with minimal prefix is challenging, especially in long files.

On answering the second research question, based on the successful application of our system to the two programming languages, we can claim that our code completion system is language-agnostic. This system can be incorporated into any programming language.

3. Related Work

There are various studies conducted up to now which use large code base and/or machine learning to code completion. One is by Svyatkovskiy et al. from Microsoft, who introduced a system named *IntelliCode Compose* [6]. This system leverages GPT-C, a variant of OpenAI's GPT-2 [5], trained on a vast dataset of program source code. It is designed to generate sequences of tokens that form syntactically correct language constructs, such as statements containing local variables, method names, and keywords, for languages including C#. Another study by [7] explored identifier completion with ranking candidates. They sought solutions to improve the efficiency of the completion process. Rather than relying on prefix matching, used in many completion systems, they introduced subsequence matching, where user-input sequences of characters are compared to names containing them, even if they are non-consecutive. Recently a study by [8] delved into method invocation and field access completion. Nguyen et al. [9] combined program analysis and langauge model for completing a partially-input statement or suggesting a statement that immediately follows the current statement if it is a complete one. Gabel et al. [10] first observed the regularity of software code mentioned above. There are infinitely many syntactically valid statements, but there are much smaller, or may even be finite, number of pracitally useful statements. Liu et al. [11] presented a non-autoregressive model for concurrently computating candidates, each of which is a line of code starting at the cursor position. 10 lines of code immediately before the current empty line is given to the completion system when programmers write code, and also 10 lines of code immediately before every line is given as training data together with the current line. They also use some information of tokens such as keywords, identifiers, operators, etc.

4. Conclusion and Future Work

In this research, we introduced a method for automatically composing prompts to the LLM using structural candidates offered by the LR-based method and assessed the method using two programming languages. Compared to the the previous work [4], this system can now suggest textual candidates rather than structural candiates. By using structural candidates in the prompts, the system can effectively instruct the LLM to exclude the bottom structural candidates for code completion.

There are many topics for future work. A few important topics are to build an IDE for usability evaluation, to measure the effectiveness of structural candidates in the prompts to the LLM, and to compare the prediction performance of our system with that of the others particularly based on the Large Language Models.

Acknowledgments

This work was supported by Innovative Human Resource Development for Local Intellectualization program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2023-RS-2023-00256629). This work was partially supported by the Korea Internet & Security Agency (KISA) - Information Security College

Support Project. Also, this work was partially supported by JSPS KAKENHI under Grant Number 23K11053.

References

[1] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Compilers — principles, techniques, and tools, 2nd edition, Addison Wesley, 2006.

[2] I. Sasano, K. Choi, A text-based syntax completion method using lr parsing, in: Proceedings of the 2021 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2021, Association for Computing Machinery, New York, NY, USA, 2021, p. 32–43. URL: https://doi.org/10.1145/3441296.3441395. doi:10.1145/3441296.3441395.

[3] I. Sasano, K. Choi, A text-based syntax completion method using lr parsing and its evaluation, Science of Computer Programming (2023) 102957. URL: https://www.sciencedirect.com/science/article/pii/S0167642323000394. doi:https://doi.org/10.1016/j.scico.2023.102957.

[4] K. Choi, S. Hwang, H. Moon, I. Sasano, Ranked syntax completion with lr parsing, in: Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing, SAC '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 1242–1251. URL: https://doi.org/10.1145/3605098.3635944. doi:10.1145/3605098.3635944.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, https://paperswithcode.com/paper/language-models-are-unsupervised-multitask, 2018.

[6] A. Svyatkovskiy, S. K. Deng, S. Fu, N. Sundaresan, Intellicode compose: Code generation using transformer, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020, Association for Computing Machinery, New York, NY, USA, 2020, p. 1433–1443. URL: https://doi.org/10.1145/3368089.3417058. doi:10.1145/3368089.3417058.

[7] S. Hu, C. Xiao, Y. Ishikawa, Scope-aware code completion with discriminative modeling, Journal of Information Processing 27 (2019) 469–478. doi:10.2197/ipsjjip.27.469.

[8] L. Jiang, H. Liu, H. Jiang, L. Zhang, H. Mei, Heuristic and neural network based prediction of project-specific api member access, IEEE Transactions on Software Engineering 48 (2022) 1249–1267. doi:10.1109/TSE.2020.3017794.

[9] S. Nguyen, T. N. Nguyen, Y. Li, S. Wang, Combining program analysis and statistical language model for code statement completion, in: Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering, ASE '19, IEEE Press, 2020, p. 710–721. URL: https://doi.org/10.1109/ASE.2019.00072. doi:10.1109/ASE.2019.00072.

[10] M. Gabel, Z. Su, A study of the uniqueness of source code, in: Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 147–156. URL: https://doi.org/10.1145/1882291.1882315. doi:10.1145/1882291.1882315.

[11] F. Liu, Z. Fu, G. Li, Z. Jin, H. Liu, Y. Hao, L. Zhang, Non-autoregressive line-level code completion, ACM Trans. Softw. Eng. Methodol. (2024). URL: https://doi.org/10.1145/3649594. doi:10.1145/3649594, just Accepted.

Faster bivariate lexicographic Gröbner bases modulo x^k

Xavier DAHAN[1]

[1] *Tohoku university, IEHE, Sendai, Japan*

Abstract

Given t bivariate polynomials $f_1, \ldots, f_t \in \mathbb{K}[x, y]$, and an integer k we report a work-in-progress to compute a minimal, not reduced, lexicographic Gröbner basis of the ideal $\langle f_1, \ldots, f_t, x^k \rangle$ in $O^{\sim}(td^2k)$, where d is an upper bound on the y-degree of the f_i's. Using the fast normal form algorithm of Schost & St-Pierre [1], this implies that we can compute its reduced Gröbner basis in $O^{\sim}(td^2k + s^2dk)$ where s is the number of polynomials contained in the output Gröbner basis. In many instances this improves the algorithm of Schost & St-Pierre [2] based on the Howell matrix normal form that runs in time $O^{\sim}(td^{\omega}k)$.

Keywords

Gröbner bases, Lexicographic order, Bivariate, Euclidean division

1. Introduction

Background Lexicographic Gröbner bases (lexGb for short) play a fundamental role when manipulating polynomial systems due to the elimination property that they are endowed with. But the lexicographic order often does not behave well with standard Gröbner bases algorithms [3], whereas the degree reverse lexicographic order has been often observed to behave the best among monomial orders when computing a Gröbner basis. This is grounded in strong theoretical evidences [4, 5]. As a result, a standard strategy to compute a zero-dimensional lexGb consists first in computing a Gröbner basis for the degree reverse lexicographic order with efficient modern algorithms like F4, F5 [6, 7] and then proceed to a change of order algorithm [8] to compute the reduced lexGb.

In the case of two variables only, the situation is different. For t polynomials $f_1, \ldots, f_t \in \mathbb{K}[x, y]$, of maximal degree d in y, and total degree d_{tot}, Schost and St-Pierre in [2], obtains a running time in $O^{\sim}(t^{\omega}d^{\omega}d_{tot})$ when at least one f_i has for leading monomial y^d. As usual ω is the exponent of matrix multiplication. This algorithm computes the Hermite normal form of a generalized Sylvester matrix of the input polynomials, extending the case $t = 2$ treated by Lazard [9, Section 5].

The same article [2] also considers computing the reduced lexGb modulo x^k, that is of the ideal $\langle f_1, \ldots, f_t, x^k \rangle$. The idea is to work in the ring $R = \mathbb{K}[x]/x^k$ and to compute the Howell normal form of a generalized Sylvester matrix of the $f_i \in R[y]$. This Howell normal form is the adaptation of the Hermite normal form for matrices of polynomials with coefficients in R. The cost can be made of $O^{\sim}(td^{\omega}k)$.

Result We consider here the more general moduli P^k, for P an irreducible polynomial in $\mathbb{K}[x]$ of degree d_P. In this paragraph we let $d_P = 1$ to simplify comparisons. Our algorithm based on Euclidean division works in $O^{\sim}(td^2k)$ and computes a minimal but not reduced Gröbner basis.

Schost & St-Pierre's normal form algorithm. The cost of reducing this minimal lexGb is due to another article of Schost & St-Pierre [1, Section 4] and is better stated in term of the output lexGb $\mathcal{H} = (h_s, \ldots, h_0)$. Assume that $\text{LM}(h_s) \prec \cdots \prec \text{LM}(h_0)$, where $\prec$ stands for the lexicographic

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
EMAIL: xdahan@gmail.com (X. DAHAN)
ORCID: 0000-0001-6042-6132 (X. DAHAN)

order with $x \prec y$. Under our setting we have $h_s = P^k$, and if we let $\deg_x(P) = 1$ like when $P = x$, then $\deg_x(h_s) = k$. Let $\deg_y(h_0) = n_0$ be the largest y-degree of the polynomials in $\mathcal{H}$, and let $s+1 = |\mathcal{H}|$ be the number of polynomials in the output. Then reducing $\mathcal{H}$ costs $O^\sim(s^2 n_0 k)$ (see [1, Prop. 4.4 & Prop. 5.1]).

Note that $s \le \min\{k, d\}$ and $n_0 \le d$. So we obtain an algorithm that computes the reduced lexGb in $O^\sim(td^2k + s^2n_0k)$, always within $O^\sim(td^2k + s^2dk)$. This is comparable or better than $O^\sim(td^\omega k)$ as soon as $s^2n_0 = O(td^\omega)$. In the case $t = O(1)$ and $s \approx d = k$, we obtain $O^\sim(d^4)$ which is worth than $O^\sim(d^{\omega+1})$. But in many cases, the asymptotic complexity is better.

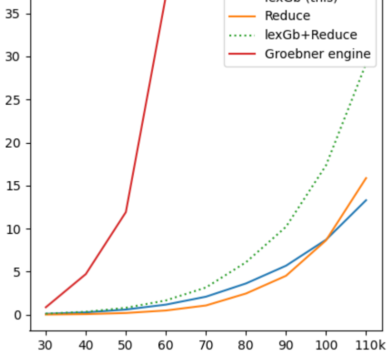
Implementation The articles [2, 1] do not mention any implementation. Indeed, implementations of the Howell normal form are apparently seldom and not available publicly — at least in major computer algebra systems. We only found Chapter 4 of the PhD thesis of Storjohann which gives the complexity of $O^\sim(d^\omega k)$, see [10, Theorem 4.6], and from which originates the result of [2].

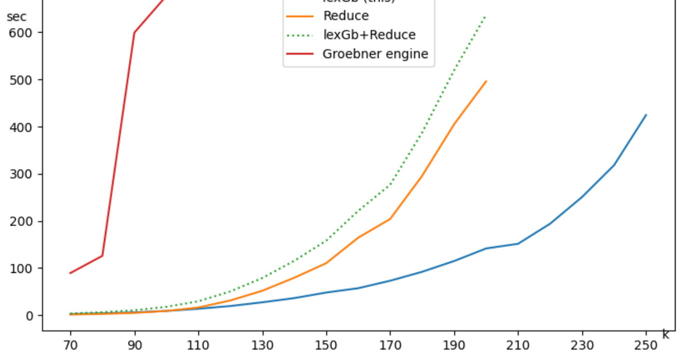
On the other hand, since the presented algorithm that computes a minimal non-reduced lexGb in $O^\sim(td^2k)$ can be compared to the quadratic-time standard Euclidean algorithm, it is efficient up to fast univariate arithmetic (polynomial operations involving the x-variable only). Our implementation in Magma (available at http://xdahan.sakura.ne.jp/lexgb24.html) supports this claim. As for the normal form algorithm modulo a reduced Gröbner basis of Schost & St-Pierre [1, Section 4], making an implementation efficient is an interesting challenge. For now we resorted here to the internal Magma command "`Reduce`" (in orange).

Some timings We tested the algorithm for two input polynomials modulo x^k:

$$a_k \equiv \left(\prod_{i=1}^{k}(y + i + x + \cdots + x^i)\right), \quad b_k \equiv (y + 1 + 2x)\left(\prod_{i=2}^{k}(y + i + x + \cdots + x^{i-1} + 2x^i)\right)$$

The reduced lexGb of $\langle a_k, b_k, x^k\rangle$ has $s + 1 = k + 1$ polynomials (the maximum possible) and is dense (it has $O(k^3)$ coefficients). Therefore, this family of examples is suitable for benchmarking as it involves worst case situations: the number of recursive calls is maximal, the cost of the normal form is maximal. Note also that taking only $t = 2$ polynomials as input is not restrictive since recursive calls involve more polynomials in the input. We report below on timings for $k = 30, 40, \ldots, 120$ (left) and $k = 70, 140, \ldots, 250$ (right) over a prime finite field of 64bits. With $t = O(1)$, $k \approx d$, the theoretical cost to obtain a minimal lexGb is $O^\sim(k^3)$. The internal command `Reduce` of Magma becomes quickly the bottleneck (in orange. Time > 500s for $k > 200$, timings are not displayed). Without surprise, its timing grows faster than the cost of the fast version of Schost & St-Pierre, which is here $O^\sim(k^4)$ (it seems to be closer to something in $O^\sim(k^5)$). Although we could not compare with the Howell form approach of [2], we could compare with





the internal Gröbner engine of Magma by calling `GroebnerBasis`$([a, b, x^k])$ (red, until $k = 100$). As already reported in [11], the timings are incomparably slow.

Scope The motivation behind working modulo P^k is twofold. Firstly, this serves as a skeleton for a similar algorithm that tackles the more general input $\langle f_1, \dots, f_t, T\rangle$ for an arbitrary polynomial $T \in \mathbb{K}[x]$, not necessarily the power of an irreducible one. See [11], which utilizes dynamic evaluation, for a detailed account when $t = 2$. Secondly, we would like to target the reduced lexGb $\mathcal{H}$ of the general input $\langle f_1, \dots, f_t\rangle$, not modulo a univariate polynomial, with our Euclidean-division based algorithm. After the work [11], a natural question asks how can we compute the lexGb of two polynomials f_1, f_2 from their subresultant sequence? To this end, it is enlightening to access the lexGbs modulo $P_i^{k_i}$, where $P_i^{k_i}$ runs over the primary factors of the elimination polynomial of the $(f_j)_j$'s. The work [12] then permits to understand how these lexGbs can reconstruct $\mathcal{H}$ via Chinese remainders. These remarks lead to a reasonable hope to compute a minimal lexGb faster than the $O^{\sim}(t^\omega d^\omega d_{tot})$ of [2].

Treating only two variables is clearly limited. Yet, all aspects shall be mastered as there is a cliff in difficulty when considering more than two variables: no general form of Lazard's structural theorem [9] which is key in this work and in [2, 1]. Let us mention though the radical case where some sort of generalizations of Lazard's theorem have been shown [13, 14, 15]. The Euclidean algorithm based approach certainly helps to understand where this difficulty stems from. One aspect of it can be related to the absence of a MonicForm routine (see Eq. (1)) that would transform a nilpotent polynomial, say in $\mathbb{K}[x, y, z]$ modulo a primary ideal in $\mathbb{K}[x, y]$. Think of $f = xz^2 + yz + x + y$, nilpotent modulo the primary $\langle x^2, y^2\rangle \subset \mathbb{K}[x, y]$. It appears that the reduced lexGb of the ideal $\langle f,\ y^2,\ x^2\rangle$ is $[f,\ y^2,\ xy,\ x^2]$. Observe the new polynomial xy introduced with the smaller variables x and y. This phenomenon does not appear for two variables only. Therefore, the Euclidean division approach helps to understand better the case of three variables or more.

Related works Recently, articles dealing with bivariate Gröbner bases have flourished. A number of them address the question of quasi-optimal asymptotic complexity estimates, with adequate genericity assumptions, and the relation with the resultant [16, 17, 18, 19, 20]. Focusing on non-generic lexGbs, the work [11] from which the present work is inspired, generalizes dynamic evaluation to a non-squarefree modulus. We have already cited [2, 1]. Besides the fast normal form in Section 4, the article [1] introduces a fast Newton iteration for general bivariate lexGbs.

2. The algorithm

Overview It is based on ideas introduced in [11], which is constrained to two input polynomials a and b. Let us summarize the content of the first part of [11] which focuses on working modulo P^k (the second part focuses on working modulo an arbitrary monic univariate polynomial). The divisions occurring in the Euclidean algorithm of a and b modulo P^k require invertible leading coefficients. In the ring $R = \mathbb{K}[x]/\langle P^k\rangle$ elements are either invertible or nilpotent. Weierstrass preparation theorem realized by Hensel lifting permits to circumvent this difficulty, by calculating a "monic form": Given $\tilde{f} \in \mathbb{K}[x, y]$ reduced modulo P^k, we denote $f, C_f \leftarrow$ MonicForm$(\tilde{f}, P^k)$ where:

$$C_f = \gcd(\text{content}(\tilde{f}),\ P^k) \in \mathbb{K}[x], \qquad f \text{ is monic in } y, \qquad \langle \tilde{f},\ P^k\rangle = \langle C_f\, f,\ P^k\rangle. \tag{1}$$

The Euclidean algorithm can be pursued with the monic f, and $C_f f$ will be part of the lexGb.

We adapt this strategy to design the main algorithm $\mathcal{H} \leftarrow$ Add$(f,\ \mathcal{G})$ where $\mathcal{G}$ is a minimal lexGb such that $\mathcal{G} \cap \mathbb{K}[x] = \langle P^{k'}\rangle$ with $k' \leq k$, $f \in \mathbb{K}[x, y]$ and $\mathcal{H}$ is a minimal lexGb of $\langle f\rangle + \langle \mathcal{G}\rangle$. Assuming for the moment this algorithm correct and running in time $O^{\sim}(d^2 k')$, the general algorithm 1 "lexGb" has the following worst-case complexity:

Algorithm 1: $\mathcal{G} \leftarrow \text{LEXGB}(f_1, \ldots, f_t; P^k)$

```
Input: Bivariate polynomials $f_1, \ldots, f_t$. Power of an irreducible polynomial $P^k \in \mathbb{K}[x]$.
Output: reduced lexGb of $\langle f_1, \ldots, f_t, P^k \rangle$
1 $f_1, \ldots, f_t \leftarrow f_1 \mod P^k, \ldots, f_t \mod P^k$          // $O^\sim(tdd_x)$ or free
2 $\mathcal{G} \leftarrow [P^k]$
3 for $i$=1,...,$t$ do
4     $\mathcal{G} \leftarrow$ ADD($f_i$, $\mathcal{G}$)          // $O^\sim(d^2kd_P)$
5 return REDUCE($\mathcal{G}$)     // REDUCE based on the normal form of [1, Section 4]. $O^\sim(s^2n_0k)$
```

Theorem 1. *Let d be the maximal degree in y of the polynomials $f_1, \ldots, f_t$. Let d_x their maximal degree in x. Let $d_P = \deg_x(P)$. Algorithm 1 computes a minimal lexGb of $\langle f_1, \ldots, f_t, P^k \rangle$ in $O^\sim(td^2kd_P + tdd_x)$.*

If the input polynomials are reduced modulo P^k, or if $P = x$ then the cost is $O^\sim(td^2kd_P)$.

The reduced lexGb requires additionally $O^\sim(s^2n_0k)$ operations in $\mathbb{K}$, where $s = |\mathcal{G}|$, $n_0 = \deg_y(g_0)$ is the largest y-degree of the polynomials in the output.

Algorithm 2: ADD(g, $\mathcal{G}$)

```
Input: $g \in \mathbb{K}[x, y]$, $\mathcal{G} = [g_0, \ldots, g_s]$ minimal lexGb modulo $P^k = g_s$
Output: minimal lexGb of $\langle g \rangle + \langle \mathcal{G} \rangle$
6 if $\mathcal{G}$ == [constant] then
7     return [1]
8 $f$, $C_f \leftarrow$ MONICFORM($g, P^k$)          // $\langle C_f\, f,\ P^k \rangle = \langle g,\ P^k \rangle$, $f$ monic, $C_f \in \mathbb{K}[x]$
9 if $f == 1$ then
10    return ADDUNIVARIATE($C_f$, $\mathcal{G}$)     //Special "easy" case where input polynomial $\in \mathbb{K}[x]$
11 if $|\mathcal{G}| == 1$ then
12    return [$C_f\, f$, $P^k$]
13 return ADDGENERIC($f$, $C_f$, $\mathcal{G}$)          // Output generates $\langle C_f\, f \rangle + \langle \mathcal{G} \rangle$
```

The main algorithm 2 "Add" The purpose is given a minimal lexGb $\mathcal{G}$ as above, not necessarily zero-dimensional, and a polynomial $f \in \mathbb{K}[x, y]$ to construct a minimal lexGb of the ideal $\langle f \rangle + \langle \mathcal{G} \rangle$. Thus, it is interesting for its own. It builds upon Euclidean divisions, the key point consists in obtaining a degree (in y) decrease through a Euclidean division (see Lines 24 and 34), and then to proceed to adequate recursive calls, with smaller input data (Lines 21 and 23). The algorithm 2 "ADD" actually only treats base cases, and then calls Algorithm 3 "ADDGENERIC", whose input are amenable to recursive calls. One base case is when $f \in \mathbb{K}[x]$ (Line 10) treated apart in the "easy" ADDUNIVARIATE. We omit this short algorithm in this work-in-progress report. Otherwise Algorithm 3 ADDGENERIC, called at Line 13, treats "generic" input: f monic, reduced modulo P^k, and $d_f := \deg_y(f) \geq 1$. Its role essentially boils down to managing four cases. Write $\mathcal{G} = [g_0, \ldots, g_s]$ ($\text{LM}(g_s) \prec \cdots \prec \text{LM}(g_0)$, so that $g_s = P^k$ and $\deg_y(g_0) = n_0$).

1. Case distinction: $\ell > k$ or $\ell \leq k$ (equivalently $C_f \nmid g_s = P^k$ or $C_f \mid g_s$)
2. Subcases distinction: $d_f \leq n_0$ or $d_f > n_0$

The first case distinction is treated by renaming variables (if-test at Line 16). The subcase distinction (if-test at Line 20) leads to call two subroutines ADDTWOA and ADDTWOB which looks very similar, but with key differences.

Algorithms AddTwoA and AddTwoB The input are monic bivariate polynomials a, b, monic univariate polynomials C_a, C_b which are powers of P, and a minimal lexGb $\mathcal{G}$ modulo P^k.

Algorithm 3: AddGeneric($f,\ C_f,\ \mathcal{G}$)

```
Input: $f \in \mathbb{K}[x,y]$ monic $\deg_y(f) \geq 1$, $C_f = P^t \in \mathbb{K}[x]$, $\mathcal{G}$ minimal lexGb modulo $P^k$
Output: minimal lexGb of $\langle C_f\, f\rangle + \langle \mathcal{G}\rangle$
14 Let $\mathcal{G} = [g_0, \dots,\ g_{s-1},\ P^k]$, and write $g_0 = M_0\, g_{0,y}$
15 $C_a \leftarrow \gcd(M_0,\ C_f)$                                   // $C_a = C_f$ or $C_a = M_0$
16 if $C_a == C_f$ then
17     $a \leftarrow f,\ b \leftarrow g_{0,y},\ C_b \leftarrow M_0$                        // $C_b b = g_0$
18 else
19     $b \leftarrow f,\ a \leftarrow g_{0,y},\ C_b \leftarrow C_f$                        // $C_a a = g_0$
20 if $\deg_y(a) > \deg_y(b)$ then        // Always holds $\langle C_a\, a,\ C_b\, b,\ P^k\rangle = \langle C_f\, f,\ g_0,\ P^k\rangle$
21     return AddTwoA($a,\ b,\ C_a,\ C_b,\ \mathcal{G}_{\geq 1}$)          // lexGb of $\langle C_a\, a,\ C_b\, b\rangle + \langle g_1, \dots, g_s\rangle$
22 else
23     return AddTwoB($a,\ b,\ C_a,\ C_b,\ \mathcal{G}_{\geq 1}$)          // lexGb of $\langle C_a\, a,\ C_b\, b\rangle + \langle g_1, \dots, g_s\rangle$
```

Additional degree constraints on a, b, C_a, C_b depend on one or the other algorithm. The output is a minimal lexGb of $\langle C_a\, a,\ C_b\, b\rangle + \langle \mathcal{G}\rangle$ (whence the name "AddTwo"). The key point is the degree decrease obtained by the Euclidean division at Lines 24 and 34. Then they undertake recursive calls. These divisions henceforth imply the complexity stated in Theorem 1.

Algorithm 4: AddTwoA($a,\ b,\ C_a,\ C_b,\ \mathcal{G}$)

```
Input: 1. $a, b \in \mathbb{K}[x,y]$ monic, $\deg_y(a) > \deg_y(b)$
2. $C_a, C_b \in \mathbb{K}[x]$ powers of $P$,   $C_a \mid C_b \mid P^k$,
3. $\mathcal{G}$ minimal lexGb modulo $P^k$, and $\deg_y(a) > \deg_y(\mathcal{G})$
Output: minimal lexGb of $\langle C_a\, a,\ C_b\, b\rangle + \langle \mathcal{G}\rangle$
24 $r \leftarrow a \mod b$     // $b$ monic. Over $R = \mathbb{K}[x]/\langle \frac{P^k}{C_b}\rangle$. It holds $\langle C_b\, a,\ C_b\, b,\ P^k\rangle = \langle C_b\, b,\ C_b\, r,\ P^k\rangle$
25 if $r \equiv 0 \mod \frac{g_1}{C_b}$ then                // Here $\langle C_b\, a,\ C_b\, b,\ P^k\rangle = \langle C_b\, b,\ P^k\rangle$
26     $\mathcal{G}'' \leftarrow$ Add($b,\ \frac{1}{C_b}\mathcal{G}$)                       // Here $\langle C_b\, \mathcal{G}''\rangle = \langle C_b\, b\rangle + \langle \mathcal{G}\rangle$
27 else
28     $\mathcal{G}' \leftarrow$ Add($r,\ \frac{1}{C_b}\mathcal{G}$)                             // $\langle C_b \mathcal{G}'\rangle = \langle C_b\, r\rangle + \langle \mathcal{G}\rangle$
29     $\mathcal{G}'' \leftarrow$ Add($b,\ \mathcal{G}'$)     // $\langle C_b \mathcal{G}''\rangle = \langle C_b\, b\rangle + \langle C_b \mathcal{G}'\rangle = \langle C_b\, b,\ C_b\, r\rangle + \langle \mathcal{G}\rangle = \langle C_b\, b,\ C_b\, a\rangle + \langle \mathcal{G}\rangle$
30 if $C_a == C_b$ then
31     return $C_b \cdot \mathcal{G}''$                                // Here $\langle C_b \mathcal{G}''\rangle = \langle C_b\, b,\ C_a\, a\rangle + \langle \mathcal{G}\rangle$
32 else
33     return $[C_a\, a]$ cat $C_b \cdot \mathcal{G}''$        // output generates $\langle C_a\, a\rangle + \langle C_b \mathcal{G}''\rangle = \langle C_a\, a,\ C_b\, b\rangle + \langle \mathcal{G}\rangle$
```

Algorithm 5: AddTwoB($a,\ b,\ C_a,\ C_b,\ \mathcal{G}$)

```
Input: 1. $a, b \in \mathbb{K}[x,y]$ monic, $\deg_y(a) \leq \deg_y(b)$,
2. $C_a,\ C_b \in \mathbb{K}[x]$ powers of $P$,   $C_a \mid C_b \mid P^k$,
3. $\mathcal{G}$ minimal lexGb modulo $P^k$,   $\deg_y(b) > \deg_y(\mathcal{G})$
Output: minimal lexGb of $\langle C_a\, a,\ C_b\, b\rangle + \langle \mathcal{G}\rangle$
34 $r \leftarrow \frac{C_b}{C_a} b \mod a$  // $a$ monic. Over $R = \mathbb{K}[x]/\langle \frac{P^k}{C_a}\rangle$. It holds $\langle C_a\, r,\ C_a\, a,\ P^k\rangle = \langle C_b\, b,\ C_a\, a,\ P^k\rangle$.
35 if $r \equiv 0 \mod \frac{P^k}{C_a}$ then                 // Here $\langle C_a\, a,\ P^k\rangle = \langle C_a\, a, C_b\, b,\ P^k\rangle$
36     return $C_a \cdot$ Add($a, \frac{1}{C_a}\mathcal{G}$)                        // Output generates $\langle C_a\, a\rangle + \langle \mathcal{G}\rangle$
37 else
38     $\mathcal{G}' \leftarrow$ Add($r,\ \frac{1}{C_a}\mathcal{G}$)  // $\langle C_a \mathcal{G}'\rangle = \langle C_a\, r\rangle + \langle \mathcal{G}\rangle$ return $C_a \cdot$ Add($a,\ \mathcal{G}'$)  // Output generates
       $\langle C_a\, a\rangle + \langle C_a \mathcal{G}'\rangle = \langle C_a\, a,\ C_b\, b\rangle + \langle \mathcal{G}\rangle$
```

References

[1] É. Schost, C. St-Pierre, Newton iteration for lexicographic Gröbner bases in two variables, Journal of Algebra 653 (2024) 325–377.

[2] É. Schost, C. St-Pierre, p-adic algorithms for bivariate Gröbner bases, in: Proceedings of the 2023 International Symposium on Symbolic and Algebraic Computation, ACM, New York, NY, USA, 2023.

[3] K. Kalorkoti, Counting and Gröbner bases, J. Symbolic Computation 31 (2001) 307–313.

[4] D. Bayer, M. Stillman, A criterion for detecting m-regularity, Inventiones mathematicae 87 (1987) 1–11.

[5] H. Loh, The converse of a theorem by Bayer and Stillman, Advances in Applied Mathematics 80 (2016) 62–69.

[6] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F4), Journal of pure and applied algebra 139 (1999) 61–88.

[7] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (F5), in: Proceedings of the 2002 international symposium on Symbolic and algebraic computation, 2002, pp. 75–83.

[8] J.-C. Faugère, P. Gianni, D. Lazard, T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, Journal of Symbolic Computation 16 (1993) 329–344.

[9] D. Lazard, Ideal bases and primary decomposition: case of two variables, Journal Symbolic Computation 1 (1985) 261–270.

[10] A. Storjohann, Algorithms for matrix canonical forms, Ph.D. thesis, ETH Zürich, 2000.

[11] X. Dahan, Lexicographic Gröbner bases of bivariate polynomials modulo a univariate one, Journal of Symbolic Computation 110 (2022) 24–65.

[12] X. Dahan, Chinese remainder theorem for bivariate lexicographic Gröbner bases, in: Proceedings of the 2023 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '23, ACM press, New York, NY, USA, 2023.

[13] M. Lederer, The vanishing ideal of a finite set of closed points in affine space, J. of Pure and Applied Algebra 212 (2008) 1116–1133.

[14] M. Marinari, T. Mora, A remark on a remark by Macaulay or enhancing Lazard structural theorem, Bull. Iranian Math. Soc. 29 (2003) 1–45, 85.

[15] B. Felszeghy, B. Ráth, L. Rónyai, The lex game and some applications, Journal of Symbolic Computation 41 (2006) 663 – 681.

[16] G. Villard, On computing the resultant of generic bivariate polynomials, in: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '18, ACM, 2018, p. 391–398.

[17] G. Villard, Elimination ideal and bivariate resultant over finite fields, in: Proceedings of the 2023 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '23, ACM press, New York, NY, USA, 2023.

[18] J. van der Hoeven, R. Larrieu, Fast reduction of bivariate polynomials with respect to sufficiently regular Gröbner bases, in: Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation, ISSAC '18, ACM, New York, NY, USA, 2018, pp. 199–206.

[19] J. van der Hoeven, R. Larrieu, Fast Gröbner basis and polynomial reduction for generic bivariate ideal, Applicable Algebra in Engineering, Communication and Computing 30 (2019) 509–539.

[20] J. van der Hoeven, G. Lecerf, Fast computation of generic bivariate resultants, Journal of Complexity 62 (2021) 101499.

Some Applications of Chinese Remainder Theorem Codes with Error-Correction

Jesse Elliott[1], Éric Schost[1]

[1]*University of Waterloo, David R. Cheriton School of Computer Science, Waterloo, Ontario, Canada*

Abstract

Modular techniques with rational reconstruction improve complexity when computing over a ground field such as $\mathbb{Q}$ by controlling the growth of intermediate expressions. Working modulo a single prime $p \in \mathbb{N}$, one can solve the problem modulo p and lift the solution to $\mathbb{Z}/p^k\mathbb{Z}$ for sufficiently large p^k using p-adic lifting techniques, when applicable. Alternatively, computations can be done modulo several small primes $p_1, \dots, p_\eta$. One can then obtain a solution modulo their product $p_1 \cdots p_\eta$ using the Chinese remainder theorem. We say primes p are "unlucky" when the procedure modulo p is not well-defined or returns a result that is different from the modulo p reduction of the rational output. Otherwise we say primes are "lucky." For some applications (solving zero-dimensional polynomial systems, for instance), testing if a prime is lucky may be prohibitively expensive. However, it is often possible to bound the number of unlucky primes by proving the existence of a nonzero $U \in \mathbb{Z}$ with all unlucky primes dividing U, and bounding the height of U. Using p-adic lifting requires the initial prime to be lucky, with a high probability of success that is determined by an upper bound on U. On the other hand, the Chinese remainder theorem requires that all primes are lucky, and to guarantee this with high probability usually requires larger primes. We report on work-in-progress that uses error correction techniques with Chinese remaindering that allows us to tolerate a few unlucky primes. Our hope is to then guarantee a high probability of success while using primes of moderate size.

We base our work on independent results from Böhm, Decker, Fieker and Pfister [1, 2] and Pernet [3]. To our knowledge, the consequences we derive, while relatively straightforward, are new. We give explicit sufficient conditions on the number of primes and their size to guarantee an arbitrary probability of success, assuming we can pick primes uniformly at random in a given interval. We also describe a number of applications.

Keywords

Chinese remainder theorem codes, polynomial system solving, modular algorithms

1. Background and previous work

Solving algebraic problems over a ground field such as $\mathbb{Q}$, considering only *algebraic complexity* (the number of base field operations) is hardly a good predictor of practical runtime: a precise analysis should take into account the size of the coefficients in the output, and the number of boolean operations throughout the execution of the algorithm.

One major challenge in such algorithms is the growth of coefficients. In many situations, we can give reasonably sharp *a priori* bounds on the bit size of the coefficients in the output

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

jakellio@uwaterloo.ca (J. Elliott); eschost@uwaterloo.ca (É. Schost)
https://uwaterloo.ca/scholar/jakellio/home (J. Elliott); https://cs.uwaterloo.ca/~eschost/ (É. Schost)

(a typical recipe being understanding them as determinants built from the input problem). Precisely, we will assume below that we know an upper bound H on the *height* of all rational numbers appearing in the output, where in this note the height $h(c)$ of a rational number c is the maximum of the base-2 logarithms of the absolute values of its minimal numerator and denominator. However, such insight should not be expected for intermediate results of our algorithm. As the algorithm progresses, the size of these coefficients may increase, before possibly collapsing when we reach the end result.

Modular techniques are used to avoid intermediate expression swell. They involve performing computations modulo one or several small primes (ideally, machine primes), thereby avoiding intermediate coefficient growth, the objective being to compute the requested output (typically, a set of polynomials, or a matrix thereof) modulo a certain large integer M. If M is large enough (precisely, if $\log_2(M) \geq H + 1$), rational reconstruction can then be applied coefficient-wise, in order to recover an output with rational coefficients.

On one end of the spectrum, one can consider working modulo a single prime p, solve the problem modulo p and lift the solution to $\mathbb{Z}/M\mathbb{Z}$, for $M = p^k$ large enough, by means of Newton / Hensel techniques, if applicable. The obvious alternative is to compute modulo sufficiently many "small" primes $(p_i)_{1\leq i\leq \eta}$ and use the Chinese remainder theorem to obtain a solution modulo $M = p_1 \cdots p_\eta$.

For most problems of interest, there exist primes p for which the procedure modulo p is not well-defined, or returns a result that differs from the modulo p reduction of the rational output; we will call these primes *unlucky*, and *lucky* otherwise. In the problems we have in mind, such as solving systems of polynomial equations, testing whether a prime is lucky may be prohibitively expensive. However, it is often possible to bound the number of unlucky primes: this is usually done by proving the existence of a nonzero $U \in \mathbb{Z}$ such that all unlucky primes divide U, and bounding the height of U.

Using p-adic lifting techniques, we need to ensure that the initial prime p is lucky; knowing the upper bound on U, we can determine what interval to pick p from in order to guarantee a high probability of success, say at least $1 - \varepsilon$ for a given tolerance ε. For Chinese remaindering algorithms, though, the direct approach requires all primes being lucky, and guaranteeing that this is the case with probability $1 - \varepsilon$ usually requires us to use larger primes (we discuss this further below). In this short note, we report on work-in-progress that uses error correction techniques (very loosely speaking, analogues of Reed-Solomon decoding, but for rational number reconstruction), where we tolerate that a few primes return wrong results (or no results at all). Our hope is then to be able to guarantee high probability of success, while using primes of moderate size.

We base our work on recent (independent) introductions of this idea, by Böhm, Decker, Fieker and Pfister [1, 2] and Pernet [3], the former in the context of algorithms for algebraic geometry, and the latter mentioning applications to linear algebra. The decoding algorithms and the sufficient conditions for success given in these two families of references are distinct, but similar; both follow an iterative reduction procedure, stated as a variant of Euclid's algorithm in Pernet's work, and as a variant of Gaussian lattice reduction by Böhm *et al.* Our presentation will follows Pernet's.

The core of our discussion concerns the reconstruction of a single rational number. We also point out that in the contexts we are interested in, algorithms usually return several such numbers (typically as coefficients of polynomials), and we can often predict that all these rationals admit a small common denominator. Taking this specificity into account would lead us toward error-tolerant vector rational number reconstruction; ideally, we could hope to reduce the number of primes by up to two, but as of now, this appears to be quite challenging.

2. Our contribution

Let us first review the key result regarding Chinese remaindering for rational reconstruction in the presence of errors. We consider a sequence of prime moduli $p_1, \dots, p_\eta$ and a rational number $r = f/g$, with $g > 0$. The goal is to recover r from a vector $(r_i)_{1 \le i \le \eta}$, where $r_i = r \bmod p_i$ for a certain number of *lucky* primes p_i. We tolerate a number of errors or missing values (e.g., for which p_i divides g), for which we write $r_i = \infty$, for a new symbol ∞; the corresponding primes are *unlucky*. Consider the following integers:

- $N = \sum_{i=1}^{\eta} \log_2(p_i)$
- $L = \sum_{1 \le i \le \eta, p_i \text{ unlucky prime}} \log_2(p_i)$.
- H is a given upper bound on the height of r, that is, $\log_2(|f|), \log_2(g) \le H$

Then, Pernet proved in [3, Lemma 2.5.4] that if $L < (N - 2H + 1)/2$, one can reconstruct r given the r_i's [3, Algorithm 2 p.38]. Böhm *et al.* proved similar results.

Although these statements are well-established, to our knowledge, the following consequences, while relatively straightforward, are new. We provide a quantitative analysis which gives explicit sufficient conditions on the number of primes and their size to guarantee an arbitrary probability of success, in a model where we assume we can pick primes uniformly at random in a given interval.

Stating this result requires us to take all primes into consideration. Thus, r and the upper bound H are as above, and to each prime $p \in \mathbb{N}$ corresponds a value $r(p)$ (possibly ∞); p is called lucky when $r(p) = r \bmod p$ and unlucky otherwise. We assume that there are finitely many unlucky primes and let U be their product. In addition to the output size bound H, we then need a bound C such that $\log_2(U) \le C$. Both H and C are problem-dependent (we discuss a few examples in the next section); once bounds on H and C are available, the following propositions apply.

In what follows, for simplicity, given an interval $\Sigma = \{\sigma, \dots, 2\sigma\}$, we assume that we can sample η primes in Σ uniformly without replacement, as long as this interval is known to contain at least η primes.

Proposition 1. *Let r, H, C be as above. For $\epsilon > 0$, let σ and η be integers such that $\sigma \ge \max(16, \frac{16C}{\epsilon}, 16H)$ and $\eta = \left\lceil \frac{4H}{\log_2(\sigma)} \right\rceil$. Select pairwise distinct primes $p_1, \dots, p_\eta$ independently and uniformly at random from the set $\Sigma = \{\sigma, \dots, 2\sigma\}$. Then, with probability at least $1 - \epsilon$, given $(r(p_1), \dots, r(p_\eta))$, one can reconstruct r.*

Proof. Let $\mathcal{D}$ denote the set $\{p \mid p \in \Sigma \text{ and } U \bmod p = 0\}$, and notice that the product $\prod_{p \in \mathcal{D}} p$ also divides U, so that in particular $\prod_{p \in \mathcal{D}} p \le U$. Each prime in Σ is at least equal to σ, so that

$\#\mathscr{D} \le \frac{C}{\log_2(\sigma)}$. On the other hand, the number of primes in Σ is at least $\frac{\sigma}{2\log_2(\sigma)}$ [4, Ex. 18.18], so that for a prime p chosen at random in Σ, $\mathbb{P}(p|U) \le \frac{(C/\log_2(\sigma))}{(\sigma/(2\log_2(\sigma)))} = \frac{2C}{\sigma}$.

Now, we choose η distinct primes $p_1, \ldots, p_\eta$ uniformly at random in Σ, and for $i = 1, \ldots, \eta$, we let X_i be the indicator variable defined as

$$X_i = \begin{cases} 1 & \text{if } p_i \mid U \\ 0 & \text{otherwise,} \end{cases}$$

so that $\mathbb{E}[X_i] = \mathbb{P}(X_i = 1) \le 2C/\sigma$. Define further $X = \sum_{i=1}^{\eta} X_i$, so that $\mathbb{E}[X] \le 2\eta C/\sigma$. Now, for any choice of η distinct primes (p_i) in Σ, the quantities N and L defined above satisfy $N = \sum_{i=1}^{\eta} \log_2(p_i) \ge \eta \log_2(\sigma)$ and $L = \sum_{1\le i\le \eta, p_i \text{ unlucky prime}} \log_2(p_i) \le \log_2(2\sigma)X$. From [3, Lemma 2.5.4] as cited above, we know that the error-tolerant rational reconstruction algorithm succeeds as soon as $L \le (N - 2H + 1)/2$, and in particular as soon as $\log_2(2\sigma)X \le \Delta = (\eta\log_2(\sigma) - 2H + 1)/2$. We will point out below that for our choice of η, Δ is positive.

Then, the probability of failure is at most $\mathbb{P}\left(\log_2(2\sigma)X > \Delta\right) = \mathbb{P}\left(X > \Delta/\log_2(2\sigma)\right)$, which by Markov's inequality is at most $\mathbb{E}[X]/\left(\Delta/\log_2(2\sigma)\right)$. We deduce

$$\mathbb{P}(\text{fail}) \le \left(\frac{2\eta C}{\sigma}\right)\left(\frac{2\log_2(2\sigma)}{\eta\log_2(\sigma) - 2H + 1}\right) \le \frac{8C}{\sigma}\frac{\eta\log_2(\sigma)}{\eta\log_2(\sigma) - 2H} = \frac{8C}{\sigma}\frac{1}{1 - \frac{2H}{\eta\log_2(\sigma)}}.$$

Now, take $\eta = \lceil 4H/\log_2(\sigma)\rceil$, so that in particular $\eta \ge 4H/\log_2(\sigma)$, and thus $2H/(\eta\log_2(\sigma)) \le 1/2$, in which case the right-most factor in the inequality above is at most 2. Besides, this choices ensures $\Delta > 0$. To summarize, in this case, we have $\mathbb{P}(\text{fail}) \le 16C/\sigma$, and this can be made less than ϵ as soon as $\sigma \ge 16C/\epsilon$.

It remains to verify that our interval Σ contains at least η primes. We know that there are at least $\sigma/2\log_2(\sigma)$ such primes, and that η is at most $4H/\log_2(\sigma) + 1$, and one checks that if $\sigma \ge 16$ and $\sigma \ge 16H$, this is indeed less than or equal to $\sigma/2\log_2(\sigma)$. ☐

Remark 2. *In the context of a modular algorithm, the most important component in the cost analysis is the total time spent solving the problem modulo the primes p_i. In rough approximation, one can assume that each such execution takes T operations modulo p_i, where T is independent of i. It follows that the total boolean cost is softly-linear in $T\sum_{1\le i\le \eta}\log_2(p_i) \in \Theta(T\eta\log(\sigma))$.*

In our construction, we have $\eta\log_2(\sigma) \le (\frac{4H}{\log_2(\sigma)} + 1)\log_2(\sigma) = 4H + \log_2(\sigma)$. In other words, the boolean cost involves both the output size H, which is as expected, together with $\log_2(\sigma)$, which will increase if we take ϵ close to zero.

Remark 3. *Assume that we do not use error-correction. In this case, in order to be able to reconstruct r, we need* all *primes to be lucky. With notation as in the proposition, we saw that the probability that a single prime is unlucky is at most $2C/\sigma$, so when choosing η primes, the probability that at least one of them is unlucky is at most*

$$1 - \left(1 - \left(\frac{2C}{\sigma}\right)\right)^{\eta} \le \frac{2\eta C}{\sigma}.$$

Assuming we choose η as above, let us derive a bound on σ that ensures $2\eta C/\sigma < \epsilon$. We proceed informally and take $\eta = 4H/\log_2(\sigma)$, so our inequality is satisfied when $8CH/(\sigma \log_2(\sigma)) < \epsilon$. Hence we require that $\sigma \log_2(\sigma) \geq 8HC/\epsilon$; this gives $\sigma \geq R(8HC/\epsilon)$, where R is the reciprocal function of $x \mapsto x \log_2(x)$. This function grows like $x/\ln(x)$, for $x \to \infty$, which gives asymptotically $\sigma \geq 8HC/(\epsilon \ln(8HC/\epsilon))$. As expected, this is inferior to the bound given in the previous proposition.

3. Applications

We end this note with a quick description of possible use cases of this work.

Computing Hermite forms over $\mathbb{Q}[x]$. In [5], Storjohann gives a modular algorithm to compute Hermite forms for matrices with entries in $\mathbb{Q}[x]$, together with bounds H on the output size and C on the unlucky primes. Our work applies directly to this situation. We are not aware of alternative methods that would rely on Newton iteration.

Computing lexicographic Gröbner bases in $\mathbb{Q}[x, y]$. In [6], St-Pierre and Schost provide similar bounds H and C for the computation of bivariate Gröbner bases; again, our work applies directly. In this case, an alternative approach based on Newton iteration exists, but has rather high complexity.

Solving zero-dimensional systems in $\mathbb{Q}[x_1, \ldots, x_n]$. The main application we have in mind is the solution of zero-dimensional polynomial systems (by means of a data-structure known as a zero-dimensional parametrization, see [7] for a definition and references). When the complex solutions have multiplicity one, a simple form of Newton iteration is applicable [8], but without this assumption, lifting techniques are complex to analyze. In this case, a bound H on the output size is available by means of the arithmetic Bézout theorem, but the unlucky primes are harder to describe. The reference [9] quantifies primes p for which the number of solutions changes modulo p, but further arguments are needed to control other possible degeneracies.

References

[1] J. Böhm, W. Decker, C. Fieker, G. Pfister, The use of bad primes in rational reconstruction, Math. Comput. 84 (2012) 3013–3027.

[2] J. Böhm, W. Decker, C. Fieker, S. Laplagne, G. Pfister, Bad primes in computational algebraic geometry, in: Mathematical Software – ICMS 2016, Springer, 2016, pp. 93–101.

[3] C. Pernet, High Performance and Reliable Algebraic Computing, HDR, Université Joseph Fourier, Grenoble 1, 2014. URL: https://theses.hal.science/tel-01094212.

[4] J. von zur Gathen, J. Gerhard, Modern Computer Algebra, third ed., Cambridge University Press, 2013.

[5] Storjohann, A., Computation of Hermite and Smith Normal Forms of Matrices, Master's thesis, University of Waterloo, 1994.

[6] E. Schost, C. St-Pierre, p-adic algorithm for bivariate gröbner bases, in: ISSAC'23, ACM, 2023, p. 508–516. doi:10.1145/3597066.3597086.

[7] E. Schost, M. S. E. Din, Bit complexity for multi-homogeneous system solving application to polynomial minimization, Journal of Symbolic Computation 87 (2018) 176–206.

[8] W. Trinks, On improving approximate results of buchberger's algorithm by newton's method, SIGSAM Bull. 18 (1984) 7–11.

[9] C. D'Andrea, A. Ostafe, I. Shparlinski, M. Sombra, Reductions modulo primes of systems of polynomial equations and algebraic dynamical systems, Trans. Amer. Math. Soc. 371 (2019) 1169–1198.

Functional Decomposition of Sparse Polynomials (Short Talk Abstract)

Mark Giesbrecht

Cheriton School of Computer Science, University of Waterloo, Canada

Keywords
Computer algebra, sparse polynomials, complexity.

We consider the algorithmic problem of functionally decomposing sparse polynomials. For example, given a (ridiculously) high degree ($5 \cdot 2^{100}$) and very sparse (7 terms) polynomial such as:

$$f(x) = x^{5\cdot 2^{100}} + 15 \cdot x^{2^{102}+2^{47}} + 90 \cdot x^{3\cdot 2^{100}+2^{48}} + 270 \cdot x^{2^{101}+3\cdot 2^{47}} + 405 \cdot x^{2^{100}+2^{49}} + 243 \cdot x^{5\cdot 2^{47}} + 1,$$

we ask how to determine quickly whether it can be written as a composition of lower degree polynomials such as

$$f(x) = g(h(x)) = g \circ h = (x^5 + 1) \circ (x^{2^{100}} + 3x^{2^{47}}),$$

and if so, to generate such a decomposition.

That such decompositions remain sparse was first conjectured for perfect powers in 1949 by Erdős [3], but not proven until 1987 by Schinzel [9]. Zannier [10] then generalized this theory to functional decompositions.

Computationally, we have had algorithms for functional decomposition of (dense) polynomials since Barton & Zippel [2] in 1976. The first polynomial-time (in the degree) algorithms appeared in 1986 by [7], at least in the "tame" case, where the characteristic of the underlying field does not divide the degree, and an almost linear time algorithm was shown later in [4]. In fact, we can now show that, except for a very specific class of polynomials, Barton & Zippel's algorithm runs in polynomial time in the degree [5]. Polynomial-time algorithms for the (dense) "wild" case and rational functions have now been been developed, most completely in [1].

Algorithms for polynomial decomposition that exploit sparsity have remained elusive until recently (see [6, 8]). We want algorithms that run in time polynomial in the *representation size* – the length/logarithm of the exponents and coefficients of the non-zero terms of the input (and output). In this talk I will present some new algorithms which meet this goal, and provide very fast and simple solutions to some polynomial decomposition problems, such as the example above. These new methods require time quadratic in the number of non-zero terms in the input and output, and in the logarithm of the degree and coefficients.

Many open algorithmic problems remain for sparse polynomials, including detecting indecomposability, the "wild" case, and rational functions. We show connections to the well-known open (and possibly intractable) problems of sparse polynomial divisibility and irreducibility. There is also considerable room to tighten bounds in the underlying mathematics (and thereby improve the cost), as well as to explore a broader class of sparsely represented functions [8].

This is ongoing work with Saiyue Liu (UBC) and Daniel S. Roche (USNA).

References

[1] L. Allem, J. Capaverde, M. van Hoeij, J. Szutkoski, Functional decomposition using principal subfields, in: Proc. 2017 ACM International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery, New York, NY, USA, 2017, pp. 421–428.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

mwg@uwaterloo.ca (M. Giesbrecht)
https://uwaterloo.ca/~mwg (M. Giesbrecht)

[2] D. Barton, R. Zippel, A polynomial decomposition algorithm, in: Proceedings of the third ACM symposium on symbolic and algebraic computation, SYMSAC '76, 1976, pp. 356–358.

[3] P. Erdős, On the number of terms of the square of a polynomial, Nieuw Arch. Wiskunde (2) 23 (1949) 63–65.

[4] J. von zur Gathen, D. Kozen, S. Landau, Functional decomposition of polynomials, in: Proc. 28th Ann. IEEE Symp. Foundations of Computer Science, Los Angeles CA, 1987, pp. 127–131.

[5] M. Giesbrecht, J. May, New algorithms for exact and approximate polynomial decomposition, in: Proc. International Workshop on Symbolic-Numeric Computation (SNC), 2005, pp. 99–112.

[6] M. Giesbrecht, D. S. Roche, Detecting lacunary perfect powers and computing their roots, Journal of Symbolic Computation 46 (2011) 1242–1259.

[7] D. Kozen, S. Landau, Polynomial Decomposition Algorithms, Technical Report 86–773, Department of Computer Science, Cornell University, Ithaca NY, 1986.

[8] S. Lyu, Faster algorithms for sparse decomposition and sparse series solutions to differential equations, Master's thesis, U. Waterloo, Waterloo, ON, Canada, 2022.

[9] A. Schinzel, On the number of terms of a power of a polynomial, Acta Arith. 49 (1987) 55–70.

[10] U. Zannier, On composite lacunary polynomials and the proof of a conjecture of Schinzel, Inventiones Mathematicae 174 (2008) 127–138.

Towards Trajectory Planning of a Robot Manipulator with Computer Algebra using Bézier Curves for Obstacle Avoidance*

Ryo Hatakeyama[1], Akira Terui[2,*] and Masahiko Mikawa[3]

[1]Master's Program in Mathematics, Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8571, Japan

[2]Institute of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan

[3]Institute of Library, Information and Media Science, University of Tsukuba, Tsukuba 305-8550, Japan

Abstract

This paper discusses the trajectory planning of a robot manipulator with computer algebra. In the operation of robot manipulators, it is important to make the trajectory of the end effector so that it does not collide with obstacles. For this purpose, we have proposed a method of generating a method of trajectory planning using cubic spline curves. However, the method has the disadvantage that the trajectory may not be included in the feasible region of the manipulator, thus an extra test for the inclusion of the curve is needed. In this paper, we propose a new method of generating a trajectory using Bézier curves, which is guaranteed to be included in the feasible region.

Keywords

Trajectory planning, Quantifier elimination, Inverse kinematics, Bézier curves

1. Introduction

In this paper, we discuss the trajectory planning of a robot manipulator with computer algebra. A manipulator is a robot with links, which correspond to human arms, and joints, which correspond to human joints, connected alternately. The end effector is the component located at the tip of the link that is farthest from the base of the manipulator. The inverse kinematics problem examines whether it is possible to place the end effector at a given point and orientation, and if it is possible, the problem is to find the joint configuration for that placement. The trajectory planning problem is to find a path for the end effector to move from the start point to the endpoint without colliding with obstacles.

There are numerous methods for solving the inverse kinematics problem of manipulators using computer algebra proposed to date [1, 2, 3, 4, 5, 6]. Among them, we have proposed one that enhances computation efficiency with the use of Comprehensive Gröbner Systems (CGS) [7], and certifies the existence of solutions to inverse kinematics problems using the Quantifier Elimination (QE) which is based on the CGS, so-called the CGS-QE [8] method [9]. Furthermore, we have extended the method to trajectory planning using a straight-line path [10].

By using the straight-line path, the trajectory may interfere with obstacles. Therefore, it is possible to design the trajectory to avoid obstacles using a polyline, but doing so may result in discontinuities in velocity and acceleration of the manipulator at the vertices of the polyline, which could destabilize the operation. To achieve smooth movement of the end effector, we have

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

*This work was partially supported by JKA and its promotion funds from KEIRIN RACE.

*Corresponding author.

s2320143@u.tsukuba.ac.jp (R. Hatakeyama); terui@math.tsukuba.ac.jp (A. Terui); mikawa@slis.tsukuba.ac.jp (M. Mikawa)

https://researchmap.jp/aterui (A. Terui); https://mikawalab.org/ (M. Mikawa)

0000-0003-0846-3643 (A. Terui); 0000-0002-2193-3198 (M. Mikawa)

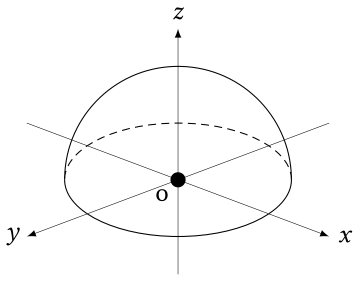


Figure 1: The feasible region of the manipulator.

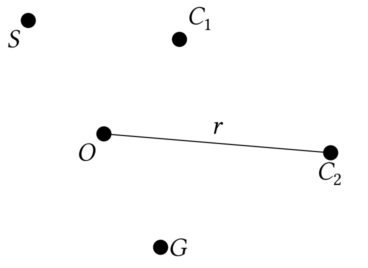


Figure 2: Defining the radius of the feasible region.

proposed trajectory planning using a spline curve [11]. However, the spline curve may not be included in the feasible region of the manipulator, thus an extra test for the inclusion of the curve is needed.

In this paper, we propose methods for trajectory planning using Bézier curves. The shape of a Bézier curve is determined by the placement of control points, and the curve has the property of always being contained within the convex hull formed by the control points as vertices. Utilizing this property, the goal is to generate trajectories that avoid obstacles without deviating from the feasible region. Although several proposals have been made for the use of Bézier curves in general trajectory planning [12, 13, 14], to the best of the authors' knowledge, the utilization of Bézier curves in trajectory planning using computer algebra is considered to be almost nonexistent.

This paper is organized as follows. In Section 2, we introduce the feasible region of the manipulator and define the radius of the region. In Section 3, we introduce the Bézier curve and its properties. In Section 4, we propose two methods for trajectory planning using Bézier curves. In Section 5, we conclude and discuss future research direction.

2. Preliminaries

In this paper, the manipulator to be used is placed on the real space $\mathbb{R}^3$ with the global coordinate system, whose origin is located at the base of the manipulator. Assume that the end effector can be placed anywhere in the feasible region except the origin (see fig. 1).

The feasible region refers to the range that the end-effector of the manipulator can reach. Here, we assume that the feasible region of the manipulator is given as the surface and the interior of a hemisphere with the origin as the center and positive z coordinates.

As for the radius of the region, we define a value different from the radius that the actual manipulator can execute. In this case, we will define the radius by comparing the distances between the origin and the start and the endpoints of the end effector, and the farthest of the passing points, as described below.

An example is shown in fig. 2. Let us consider a trajectory with the start point S and the endpoint G that passes through two predetermined points C_1 and C_2. Assume that we have verified the end-effector can reach all the points except for the origin O by, for example, the method we have previously proposed [10]. S, C_1, C_2, and G are located as in fig. 2 with respect to the origin O. In this case, the point farthest from O among these is C_2. Therefore, the distance r between O and C_2 is defined as the r, radius of the feasible region. We see that, by the radius r in this way, the end-effector will of course be able to reach a point closer than the point where the end-effector is already determined to be placed.

3. The Bézier Curve

The Bézier curve [15] is a parametric curve on which the point is expressed as a polynomial function of the parameter t, defined as follows.

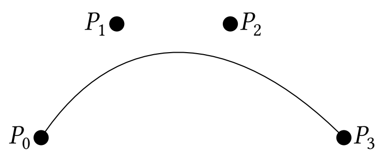


Figure 3: Construction of a 3-degree Bézier Curve.

Definition 1. Let $P_0, P_1, \dots, P_n$ be different points. Then, the *N*-degree Bézier Curve $P(t)$ is defined as

$$P(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} P_i, \quad 0 \le t \le 1,$$

where $P_0, P_1, \dots, P_n$ are the control points. Note that the curve starts at P_0 and ends at P_n.

The method employs control points for producing the curve. Note that the curve does not pass through the control points except for the start and the endpoint but is attracted by them. It is as if the points exert a pull on the curve.

An *n*-degree Bézier Curve is constructed as follows (for an example of a 3-degree Bézier curve, see Figure 3). First, place $n+1$ control points $P_0, P_1, \dots, P_n$, and take points $Q_0^{(1)}, Q_1^{(1)}, \dots, Q_{n-1}^{(1)}$ that divide the line segment $P_0P_1, P_1P_2, \dots, P_{n-1}P_n$ internally into $t : 1-t$, respectively. Next, take points $Q_0^{(2)}, Q_1^{(2)}, \dots, Q_{n-2}^{(2)}$ that divide the line segments $Q_0^{(1)}Q_1^{(1)}, Q_1^{(1)}Q_2^{(1)}, \dots, Q_{n-2}^{(1)}Q_{n-1}^{(1)}$, internally into $t : 1-t$, respectively. Repeat the same operation for n times to obtain $Q_0^{(n)}$, then the locus of is $Q_0^{(n)}$ for $0 \le t \le 1$ constructs the Bézier Curve.

One of the advantages of using Bézier Curves is the convex hull property of the curves [13]. For the convex hull $A = \{\sum_{i=0}^{n} c_i P_i \mid \sum_{i=0}^{n} c_i = 1, 0 \leqq c_i \leqq 1\}$ of the control points $P_0, P_1, \dots, P_n$, we see that any point on the Bézier Curve $P(t)$ is included in A. In other words, if a polyhedron with each control point as a vertex is included in the feasible region, the Bézier Curve obtained from those control points is also included in the region. This idea will be used in our second method proposed below.

4. Path planning using Bézier curves

We propose two methods for trajectory planning using Bézier curves. In the methods, we settle the following assumptions: In addition to the start and the endpoints, two points that the curve must pass through are given. Those points have the same y and z coordinates, respectively.

4.1. Method 1: path planning with single Bézier curve

In the first method, we construct a path with a single Bézier curve. We give the x coordinates of the control points equally distributed and select the control points using equality and inequality evaluation. Furthermore, we consider only the case where the curve is curved in the positive direction in both y and z coordinates.

If an *n*-degree Bézier Curve in $\mathbb{R}^3$ is represented as $P(t) = (P_x(t), P_y(t), P_z(t))$, each component is expressed as

$$P_x(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} x_i, \quad P_y(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} y_i, \quad P_z(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} z_i, \tag{1}$$

using the control points $P_i(x_i, y_i, z_i)$ $(i = 0, 1, \dots n)$. Here, we define the x coordinates of each control point satisfying that $P_x(t)$ is a linear polynomial in t. This approach has the advantage that it not only reduces the amount of calculation required to create the trajectory but also makes the operation of "finding the value of t from the x coordinate of a point on the curve" easier.

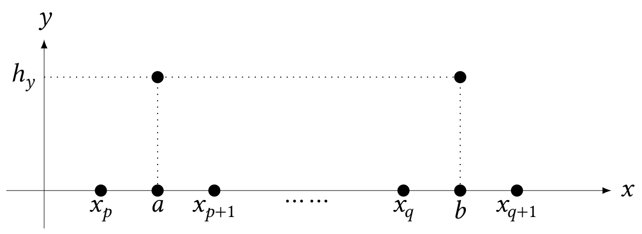


Figure 4: Position of (a, h_y) and (b, h_y).

To achieve this objective, simply place the x coordinates $x_0, x_1, \dots, x_n$ of each control point in order at equal intervals, i.e., for $i = 0, 1, \dots, n$, let

$$x_i = \frac{(n-i)x_0 + ix_n}{n}.$$

Substituting x_i into $P_x(t)$ in eq. (1) gives

$$P_x(t) = (1-t)x_0 + tx_n. \tag{2}$$

Next, we define the y and z coordinates of each control point. In eq. (2), denote the x coordinate of a point on the curve simply as $P_x(t) = x$, and solve eq. (2) for t, then we have

$$t = \frac{x - x_0}{x_n - x_0}. \tag{3}$$

Now, assume that there is an obstacle in the feasible region, and to avoid it, the y and z coordinates of the path must always exceed a certain value $h_y > 0$ and $h_z > 0$, respectively, in the interval $[a, b]$ in the x coordinate, where $x_p < a \le x_{p+1} \le x_q \le b < x_{q+1}$ for $1 \le p < q + 1 \le n$ (see Figure 4). Substituting $x = a$ and b for eq. (3) and denoting them as t_a and t_b, respectively, we obtain

$$t_a = \frac{a - x_0}{x_n - x_0}, \quad t_b = \frac{b - x_0}{x_n - x_0}.$$

For the y coordinate of the curve, among the tuples of $y_1, y_2, \dots y_{n-1}$ that satisfy the following equalities and inequalities:

$$P_y(t_a) = P_y(t_b) = h_y, \quad y_0 \le y_1, \dots, y_p, \quad h_y \le y_{p+1}, \dots, y_q, \quad y_n \le y_{q+1}, \dots, y_{n-1}, \tag{4}$$

select the one with the smallest sum $\sum_{k=p+1}^{q} y_k$. The reason for selecting the tuple with the smallest sum is to minimize the bulge in the positive direction of the y coordinate of the curve. Furthermore, to prevent each y_i value from becoming too small, a lower limit is set as constraints in eq. (4) so that it does not fall below the coordinate of the starting point before crossing the obstacle, and the coordinate of the endpoint after crossing the obstacle.

For the z coordinate of the curve, $z_1, z_2, \dots, z_{n-1}$ can be calculated in the same way as $y_1, y_2, \dots y_{n-1}$ are calculated in above, simply by replacing y_i with z_i. After obtaining $y_1, y_2, \dots y_{n-1}$ and $z_1, z_2, \dots, z_{n-1}$, put these values into $P_y(t)$ and $P_z(t)$ in eq. (1), respectively, which gives the coordinates of all control points, thus the curve becomes the trajectory of the end effector.

The advantage of this method is that the x coordinate of a point on the curve is expressed as a linear polynomial t, thus the value of t can easily obtained from x as in eq. (3), and the curve can be flexibly designed according to the position and the size of the obstacles. However, at present, it is not guaranteed that the curve will be included in the feasible region, thus improvements of the method are required, such as not only suppressing the bulge of the curve but also adding new constraints so that the curve will be included in the feasible region.

4.2. Method 2: path planning with multiple Bézier curves

The second method is to construct a path connecting multiple 3-degree Bézier curves calculated under certain conditions. We use a Bézier curve of degree 3 because it is the curve of the smallest degree whose curvature can be controlled. This method makes effective use of the convex hull property mentioned above since it is guaranteed that the created curves do not go outside the feasible region.

We consider the connection of three cubic Bézier Curves $P(t), Q(t)$, and $R(t)$, where

$$P(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} P_i, \quad Q(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} Q_i, \quad R(t) = \sum_{i=0}^{3} \binom{3}{i} t^i (1-t)^{3-i} R_i. \tag{5}$$

Assume that, in eq. (5), $P_i, Q_i, R_i \in \mathbb{R}^3$ and we consider $P(t), Q(t), R(t)$ as vector-valued polynomials. As with the method in Section 4.1, we aim to draw a curve in which the y and the z coordinates exceed a certain value in a certain interval in the x coordinate, by drawing the first curve $P(t)$ from the starting point to that interval, the second curve $Q(t)$ avoiding the obstacles, and the third curve to the endpoint.

Since the endpoint of $P(t)$ and the start point of $Q(t)$, and the end point of $Q(t)$ and the start point of $R(t)$ are identical, respectively, we have $P(1) = Q(0)$ and $Q(1) = R(0)$, i.e., $P_3 = Q_0$ and $Q_3 = R_0$. In this path planning, we aim to express the other control points using the predefined points P_0, Q_0, R_0, R_3.

To make the connection smooth, we settle a constraint that the first and second derivatives at the connection points are equal for two adjacent curves, respectively. The necessary and sufficient conditions for $P'(1) = Q'(0)$ and $Q'(1) = R'(0)$ to hold are $2Q_0 = P_2 + Q_1$, $2R_0 = Q_2 + R_1$, respectively. Similarly, the necessary and sufficient conditions for $P''(1) = Q''(0)$ and $Q''(1) = R''(0)$ to hold are $2(Q_1 - P_2) = Q_2 - P_1, \quad 2(R_1 - Q_2) = R_2 - Q_1$, respectively [16]. So far, we see that the control points P_1, P_2, R_1, R_2 depend on Q_1 and Q_2 as

$$P_1 = 4Q_0 - 4Q_1 + Q_2, \quad P_2 = 2Q_0 - Q_1, \quad R_1 = 2R_0 - Q_2, \quad R_2 = Q_1 - 4Q_2 + 4R_0. \tag{6}$$

Now we determine Q_1 and Q_2. Let the feasible region be a hemisphere with a radius $r = \max\{\|P_0\|, \|Q_0\|, \|R_0\|, \|R_3\|\}$ centered at the origin and with $z > 0$. Here, $\|\cdot\|$ represents the norm of the vector. We require all remaining control points to be included in this hemisphere, i.e.,

$$\|P_i\| \leq r, \quad \|Q_i\| \leq r, \quad \|R_i\| \leq r \quad (i = 1, 2),$$

which is reduced to solve the problem to find Q_1 and Q_2 that satisfy the above conditions. This problem is expressed as eliminating quantified variables in a quantified formula

$$\begin{aligned} \exists P_1 \exists P_2 \exists R_1 \exists R_2 ((P_1 = 4Q_0 - 4Q_1 + Q_2) \wedge (P_2 = 2Q_0 - Q_1) \wedge (R_1 = 2R_0 - Q_2) \wedge (R_2 = Q_1 - 4Q_2 + 4R_0) \\ \wedge (\|P_1\| \leqq r) \wedge (\|P_2\| \leqq r) \wedge (\|Q_1\| \leqq r) \wedge (\|Q_2\| \leqq r) \wedge (\|R_1\| \leqq r) \wedge (\|R_2\| \leqq r)). \end{aligned} \tag{7}$$

After solving eq. (7) and conditions on Q_1 and Q_2 are obtained, then choose Q_1 and Q_2 that makes $\|Q_1 - Q_0\|^2 + \|Q_2 - Q_1\|^2 + \|R_0 - Q_2\|^2$ as small as possible satisfying eq. (7), Then the rest of the control points $P_0, P_1, P_2, Q_0, R_0, R_1, R_2, R_3$ are determined, and the path of the end effector is obtained.

This method is promising since, if eq. (7) is solved, it gives a path that does not go beyond the feasible region and passes two points through for obstacle avoidance. Unfortunately, quantifier elimination for eq. (7) is computationally expensive and, at present, has not yet been successful. Therefore, by relaxing the original problem, we propose an alternative method to define a path in practical time.

4.2.1. Using 2-degree Bézier curves

In the alternative method, we create a path using three 2-degree Bèzier Curves

$$\hat{P}(t) = \sum_{i=0}^{2} \binom{2}{i} t^i (1-t)^{2-i} \hat{P}_i, \quad \hat{Q}(t) = \sum_{i=0}^{2} \binom{2}{i} t^i (1-t)^{2-i} \hat{Q}_i, \quad \hat{R}(t) = \sum_{i=0}^{2} \binom{2}{i} t^i (1-t)^{2-i} \hat{R}_i,$$

in place of $P(t), Q(t), R(t)$ in eq. (5), respectively, whose first-order differentials are identical at the connection points, together with $\hat{P}_2 = \hat{Q}_0, \hat{Q}_2 = \hat{R}_0$. Now we determine the other control points using only the predefined $\hat{P}_0 = P_0, \hat{Q}_0 = Q_0, \hat{R}_0 = R_0, \hat{R}_2 = R_3$. Since the first derivatives at the connection points are the same for two adjacent curves, we have $2\hat{Q}_0 = \hat{P}_1 + \hat{Q}_1, 2\hat{R}_0 = \hat{Q}_1 + \hat{R}_1$. For $r := \max\{\|\hat{P}_0\|, \|\hat{Q}_0\|, \|\hat{R}_0\|, \|\hat{R}_3\|\}$, we require all remaining control points to be included in the hemisphere centered at the origin with radius r, i.e., $\|\hat{P}_1\| \le r, \|\hat{Q}_1\| \le r, \|\hat{R}_1\| \le r$. Then, the problem is reduced to eliminate quantified variables in a quantified formula

$$\exists \hat{P}_1 \exists \hat{R}_1 ((\hat{P}_1 = 2\hat{Q}_0 - \hat{Q}_1) \wedge (\hat{R}_1 = 2\hat{R}_0 - \hat{Q}_1) \wedge (\|\hat{P}_1\| \le r) \wedge (\|\hat{Q}_1\| \le r) \wedge (\|\hat{R}_1\| \le r)). \tag{8}$$

Fortunately, by quantifier elimination, we have obtained the possible region of $\hat{Q}_1$ that satisfies eq. (8), then three Bézier Curves were obtained in the same way as described in the cubic Bézier Curves case. (The computation was done using the computer algebra system Wolfram Mathematica 13.3.1 in approximately 15.4 [s]. The computing environment is as follows: Intel Core i3-8130U 2.20GHz, 8GB RAM, Windows 11 Home.)

5. Concluding Remarks

We have proposed two methods for trajectory planning of manipulators using Bézier curves so that the generated curve does not go outside the feasible region. The first method can avoid obstacles with a minimum curve that matches the position and size of the obstacle, but it does not guarantee that any point on the curve is included in the region. The second method can guarantee that the curve does not go outside the region based on the constraint conditions, but its calculation cost is high and the position of the control point has not yet been obtained. To overcome this, we have proposed an alternative method using 2-degree Bézier curves, which can be calculated in a practical time.

Our future work includes the establishment of a better method for selecting the control points of a curve that overcomes current issues. We believe that method 2 is promising since the curve generated by the method is guaranteed to be included within the feasible region. The next issue to be solved is to improve computational efficiency by deriving more appropriate constraints, making the algorithm more efficient, and so on.

Once this method is established, we will move on to the stage of calculating the sequence of joint placements for the completed trajectory, leading to improved trajectory planning as proposed in our previous work [10].

References

[1] J. Capco, M. S. E. Din, J. Schicho, Robots, computer algebra and eight connected components, in: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation, ISSAC '20, ACM, 2020. doi:10.1145/3373207.3404048.

[2] J.-C. Faugère, J.-P. Merlet, F. Rouillier, On solving the direct kinematics problem for parallel robots, Research Report RR-5923, INRIA, 2006. URL: https://hal.inria.fr/inria-00072366.

[3] C. M. Kalker-Kalkman, An implementation of Buchbergers' algorithm with applications to robotics, Mech. Mach. Theory 28 (1993) 523–537. doi:10.1016/0094-114X(93)90033-R.

[4] S. Ricardo Xavier da Silva, L. Schnitman, V. Cesca Filho, A Solution of the Inverse Kinematics Problem for a 7-Degrees-of-Freedom Serial Redundant Manipulator Using Gröbner Bases Theory, Mathematical Problems in Engineering 2021 (2021) 6680687. doi:10.1155/2021/6680687.

[5] T. Uchida, J. McPhee, Triangularizing kinematic constraint equations using Gröbner bases for real-time dynamic simulation, Multibody System Dynamics 25 (2011) 335–356. doi:10.1007/s11044-010-9241-8.

[6] T. Uchida, J. McPhee, Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms, Mech. Mach. Theory 52 (2012) 144–157. doi:10.1016/j.mechmachtheory.2012.01.015.

[7] V. Weispfenning, Comprehensive Gröbner Bases, J. Symbolic Comput. 14 (1992) 1–29. doi:10.1016/0747-7171(92)90023-W.

[8] R. Fukasaku, H. Iwane, Y. Sato, Real Quantifier Elimination by Computation of Comprehensive Gröbner Systems, in: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 173–180. doi:10.1145/2755996.2756646.

[9] S. Otaki, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Real Quantifier Elimination based on Comprehensive Gröbner Systems, Preprint, 2021. doi:10.48550/arXiv.2111.00384, arXiv:2111.00384.

[10] M. Yoshizawa, A. Terui, M. Mikawa, Inverse Kinematics and Path Planning of Manipulator Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems, in: Computer Algebra in Scientific Computing: CASC 2023, volume 14139 of Lecture Notes in Computer Science, Springer, 2023, pp. 393–419. doi:10.1007/978-3-031-41724-5_21.

[11] Y. Shirato, N. Oka, A. Terui, M. Mikawa, An Optimized Path Planning of Manipulator with Spline Curves Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems, in: SCSS 2024 Work-in-progress Proceedings, Open Publishing Association, 2024. To appear.

[12] U. Dincer, M. Cevik, Improved trajectory planning of an industrial parallel mechanism by a composite polynomial consisting of bëzier curves and cubic polynomials, Mechanism and Machine Theory 132 (2019) 248–263. doi:10.1016/j.mechmachtheory.2018.11.009.

[13] Y.-L. Kuo, C.-C. Lin, Z.-T. Lin, Dual-optimization trajectory planning based on parametric curves for a robot manipulator, International Journal of Advanced Robotic Systems 17 (2020) 1729881420920046. doi:10.1177/1729881420920046.

[14] Z. Xu, S. Wei, N. Wang, X. Zhang, Trajectory planning with bezier curve in cartesian space for industrial gluing robot, in: X. Zhang, H. Liu, Z. Chen, N. Wang (Eds.), Intelligent Robotics and Applications, Springer International Publishing, Cham, 2014, pp. 146–154. doi:10.1007/978-3-319-13963-0_15.

[15] P. Bézier, The Mathematical Basis of the UNIURF CAD System, Elsevier, 1986. doi:10.1016/C2013-0-01005-5.

[16] T. Tsuchihashi, Trajectory Generation and Control for Manipulator Using the Bezier Curve (in Japansese), Transactions of the Japan Society of Mechanical Engineers Series C 55 (1989) 124–128. doi:10.1299/kikaic.55.124.

Algebraic (non) Relations Among Polyzetas

V. Hoang Ngoc Minh[1]

[1]University of Lille, 1 Place Déliot, Lille, 59024, France

Abstract

Two confluent rewriting systems in noncommutatives polynomials are constructed using the equations allowing the identification of the local coordinates (of second kind) of the graphs of the ζ polymorphism as being (shuffle or quasi-shuffle) characters and bridging two algebraic structures of polyzetas.

In each system, the left side of each rewriting rule corresponds to the leading monomial of the associated homogeneous in weight polynomial while the right side is canonically represented on the algebra generated by irreducible terms which encode an algebraic basis of the algebra of polyzetas.

These polynomials are totally lexicographically ordered and generate the kernels of the ζ polymorphism meaning that the free algebra of polyzetas is graded and the irreducible polyzetas are transcendent numbers, algebraically independent.

Keywords

Polylogarithms, Hamonic Sums, Polyzetas, Rewritting Systems

1. Introduction

For any $r \geq 1$ and $(s_1, \ldots, s_r) \in \mathbb{N}_{\geq 1}$, for any $z \in \widetilde{\mathbb{C} \setminus \{0,1\}}$ and $n \geq 1$, let

$$\mathrm{Li}_{s_1,\ldots,s_r}(z) := \sum_{n_1 > \ldots > n_r > 0} \frac{z^{n_1}}{n_1^{s_1} \ldots n_r^{s_r}} \quad \text{and} \quad \mathrm{H}_{s_1,\ldots,s_r}(n) := \sum_{n_1 > \ldots > n_r > 0}^{n} \frac{1}{n_1^{s_1} \ldots n_r^{s_r}}. \tag{1}$$

which are respectively called *polylogarithm* and *harmonic sum.*

Let $\mathcal{H}_r$ be $\{(s_1, \ldots, s_r) \in \mathbb{N}^r_{\geq 1}, s_1 > 1\}$. Then, for any $(s_1, \ldots, s_r)$ belonging to $\mathcal{H}_r$, by a Abel's theorem, the following limits exist and are called *polyzetas*[1] [9, 10]

$$\zeta(s_1, \ldots, s_r) := \lim_{z \to 1} \mathrm{Li}_{s_1,\ldots,s_r}(z) = \lim_{n \to +\infty} \mathrm{H}_{s_1,\ldots,s_r}(n) = \sum_{n_1 > \ldots > n_r > 0} n_1^{-s_1} \ldots n_r^{-s_r}. \tag{2}$$

Euler earlier studied polyzetas, in particular $\{\zeta(s_1, s_2)\}^{r \geq 1}_{s_1 > 1, s_2 \geq 1}$ in classic analysis. He stated that $\zeta(6, 2)$ can not be expressed on $\zeta(2)$, ..., $\zeta(8)$ and proved [6]

$$\zeta(2,1) = \zeta(3) \quad \text{and} \quad \zeta(s,1) = \frac{1}{2}\Big(s\zeta(s+1) - \sum_{j=1}^{s-2} \zeta(j+1)\zeta(s-j)\Big), s > 1. \tag{3}$$

The $\{\zeta(s_1, \ldots, s_r)\}^{r \geq 1}_{s_1 > 1, s_2, \ldots, s_r \geq 1}$ are also called *multi zeta values* (MZV for short) [13] or *Euler-Zagier sums* [2] and the numbers r and $s_1 + \ldots + s_r$ are, respectively, *depth* and *weight* of $\zeta(s_1, \ldots, s_r)$. One can also found in their biographies some recent applications of these special values in algebraic geometry, Diophantine equations, knots invariants of Vassiliev-Kontsevich, modular forms, quantum electrodynamic, ….

Many new linear relations for polyzetas are detected using LLL type algorithms in high performance computing and the truncations of $\{\zeta(s_1, \ldots, s_r)\}^{r \geq 1}_{s_1 > 1, s_2, \ldots, s_r \geq 1}$, *i.e.* $\{\mathrm{H}_{s_1,\ldots,s_r}(n)\}^{r \geq 1}_{s_1 > 1, s_2, \ldots, s_r \geq 1}$ [1, 2]. In this approach, the main problem is to detect with near certainty which polyzetas can not be

SCSS 2024: 10th International Symposium on Software Science, August 28–30, 2024, Tokyo, Japan
vincel.hoang-ngoc-minh@univ-lille.fr (V. Hoang Ngoc Minh)
0000-0002-3510-7639 (V. Hoang Ngoc Minh)

[1]Polyzeta is the contraction of polymorphism and of zeta (see (5)–(6) below).

expressed on $\{\zeta(2),\ldots,\zeta(s+k)\}$ and are qualified as *new constants* (as for Euler's $\zeta(6,2)$) [2]. Such polyzetas could be $\mathbb{Q}$-algebraically independent on these zeta values (see Example 1 below) and the polyzetas could be transcendent numbers (see [9, 10] for proof). Checking linear relations among $\{\zeta(s_1,\ldots,s_r)\}^{r\geq 1}_{\substack{s_1>1,s_2,\ldots,s_r\geq 1\\ 2\leq s_1+\ldots+s_r\leq 12}}$, Zagier stated that the $\mathbb{Q}$-module generated by MZV is graded (see [9, 10] for proof) and guessed (see [7, 8, 12] for other algebraic checks)

Conjecture 1 ([13]). *Let* $d_k := \dim \mathcal{Z}_k$ *and* $\mathcal{Z}_k := \operatorname{span}_{\mathbb{Q}}\{\zeta(w)\}^{r\geq 1}_{\substack{s_1>1,s_2,\ldots,s_r\geq 1\\ s_1+\ldots+s_r=k}}$, *for* $k\geq 1$. *Then* $d_1=0, d_2=d_3=1$ *and* $d_k=d_{k-3}+d_{k-2}$, *for* $k\geq 4$.

Studying Conjecture 1, in continuation with [3, 5] by a symbolic approach, this work provides more explanations and consequences regarding the algorithm **LocalCoordinateIdentification**, partially implemented in [3] and briefly described in [4].

It applies an Abel like theorem concerning the generating series of $\{\mathrm{H}_{s_1,\ldots,s_r}\}^{r\geq 1}_{s_1,\ldots,s_r\geq 1}$ (resp. $\{\mathrm{Li}_{s_1,\ldots,s_r}\}^{r\geq 1}_{s_1,\ldots,s_r\geq 1}$) [5], over the alphabet $Y=\{y_k\}_{k\geq 1}$ (resp. $X=\{x_0,x_1\}$) generating the free monoid $(Y^*,1_{Y^*})$ (resp. $(X^*,1_{X^*})$) with respect to the concatenation (denoted by conc and omitted when there is no ambiguity), the set of Lyndon words $\mathcal{L}ynY$ (resp. $\mathcal{L}ynX$) and the set of polynomials, $\mathbb{Q}\langle Y\rangle$ (resp. $\mathbb{Q}\langle X\rangle$). This theorem exploits the indexations of polylogarithms and harmonic sums in (1) by words, *i.e.* [9, 10]

$$\mathrm{Li}_{x_0^r}(z)=\log^r(z)/r!,\quad \mathrm{Li}_{x_0^{s_1-1}x_1\ldots x_0^{s_r-1}x_1}=\mathrm{Li}_{s_1,\ldots,s_r},\quad \mathrm{H}_{y_{s_1}\ldots y_{s_r}}=\mathrm{H}_{s_1,\ldots,s_r}. \tag{4}$$

It follows that the isomorphism of algebras $\mathrm{H}_\bullet : (\mathbb{Q}\langle Y\rangle, \,⧢\,) \longrightarrow (\mathbb{Q}\{\mathrm{H}_w\}_{w\in Y^*},\times)$ (resp. $\mathrm{Li}_\bullet : (\mathbb{Q}\langle X\rangle, ⧢) \longrightarrow (\mathbb{Q}\{\mathrm{Li}_w\}_{w\in X^*},\times)$), mapping u (resp. v) to H_u (resp. Li_v), induce the following *surjective* polymorphism [9, 10]

$$\zeta : \begin{matrix}(\mathbb{Q}1_{X^*}\oplus x_0\mathbb{Q}\langle X\rangle x_1, ⧢, 1_{X^*})\\ (\mathbb{Q}1_{Y^*}\oplus (Y\setminus\{y_1\})\mathbb{Q}\langle Y\rangle, \,⧢\,, 1_{Y^*})\end{matrix} \longrightarrow\!\!\!\!\rightarrow (\mathcal{Z},\times,1), \tag{5}$$

$$\begin{matrix}x_0x_1^{s_1-1}\ldots x_0x_1^{s_k-1}\\ y_{s_1}\ldots y_{s_k}\end{matrix} \longmapsto \zeta(s_1,\ldots,s_r), \tag{6}$$

where $\mathcal{Z}$ is the $\mathbb{Q}$-algebra generated by polyzetas (not linearly free [13]) and the product $⧢$ (resp. $⧢$) is defined, for any $u,v,w\in Y^*$ (resp. X^*) and $y_i,y_j\in Y$ (resp. $x,y\in X$), by

$$w \,⧢\, 1_{Y^*}=1_{Y^*} \,⧢\, w=w \text{ and } y_iu \,⧢\, y_jv=y_i(u \,⧢\, y_jv)+y_j(y_iu \,⧢\, v)+y_{i+j}(u \,⧢\, v), \tag{7}$$

$$(\text{resp. } w ⧢ 1_{X^*}=1_{X^*} ⧢ w=w \text{ and } xu ⧢ yv=x(u ⧢ yv)+y(xu ⧢ v)). \tag{8}$$

The graphs of the ζ polymorphism in (5)–(6) are expressed as $⧢$ (resp. $⧢$)-group like series as follows [9, 10]

$$Z_\gamma=e^{\gamma y_1}\prod_{l\in\mathcal{L}ynY\setminus\{y_1\}}^{\searrow} e^{\zeta(\Sigma_l)\Pi_l} \quad\text{and}\quad Z_{⧢}=\prod_{l\in\mathcal{L}ynX\setminus X}^{\searrow} e^{\zeta(S_l)P_l}, \tag{9}$$

where $\{\Pi_w\}_{w\in Y^*}$ (resp. $\{P_w\}_{w\in X^*}$) is the PBW-Lyndon basis (of the Lie polynomilas $\{\Pi_l\}_{l\in\mathcal{L}ynY}$ (resp. $\{P_l\}_{l\in\mathcal{L}ynX}$) basis) in duality with $\{\Sigma_w\}_{w\in X^*}$ (resp. $\{S_w\}_{w\in X^*}$) (containing the basis $\{\Sigma_l\}_{l\in\mathcal{L}ynY}$ (resp. $\{S_l\}_{l\in\mathcal{L}ynY}$)), on the $⧢$ (resp. $⧢$)-bialgebra [9, 10]. Finally, the identification of their local coordinates (of second kind in the group of group like series) in the equations bridging the lagebraic structures of polyzetas, *i.e.* [5]

$$Z_\gamma=e^{\gamma y_1-\sum_{k\geq 2}\zeta(k)(-y_1)^k/k}\pi_Y Z_{⧢} \quad\text{and}\quad Z_{⧢}=e^{-\gamma x_1+\sum_{k\geq 2}\zeta(k)(-x_1)^k/k}\pi_X Z_\gamma, \tag{10}$$

provides the algebraic relations among $\{\zeta(\Sigma_l)\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$ (resp. $\{\zeta(S_l)\}_{l\in\mathcal{L}ynX\setminus X}$), independent on γ, leading to the algebraic bases for $\operatorname{Im}\zeta$ and the homogenous polynomials generating $\ker\zeta$ [9, 10] (see [3] for examples), with[2] the morphism of monoids $\pi_Y : X^*x_1 \longrightarrow Y^*$ (resp. $\pi_X : Y^* \longrightarrow X^*x_1$) maps y_k to $x_0^{k-1}x_1$ (resp. $x_0^{k-1}x_1$ to y_k).

[2]There are one-to-one correspondences over the above monoids and that generated by $\mathbb{N}_{\geq 1}$, *i.e* $x_0^{s_1-1}x_1\ldots x_0^{s_r-1}x_1 \in X^*x_1 \rightleftharpoons^{\pi_Y}_{\pi_X} y_{s_1}\ldots y_{s_r}\in Y^* \leftrightarrow (s_1,\ldots,s_r)\in\mathbb{N}^*_{\geq 1}$

2. Rewriting among $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$ and among $\{S_l\}_{l\in\mathcal{L}ynX\setminus X}$

For convenience, $\mathcal{X}$ denotes X or Y and if $\mathcal{X}=X$ then gDIV $=X$ and CONV $=x_0X^*x_1$ else gDIV $=\{y_1\}$ and CONV $=(Y\setminus\{y_1\})Y^*$. It follows that $\mathcal{L}yn\mathcal{X}\setminus$ gDIV $\subset$ CONV.

Expressing, *i.e* replacing "=" by "→", the relations among polyzetas in [3] become the rewriting rules among polyzetas and yield the following increasing sets of irreducible polyzetas (see Example 1 below)

$$\mathcal{Z}_{irr}^{\mathcal{X},\le 2}\subset\cdots\subset\mathcal{Z}_{irr}^{\mathcal{X},\le p}\subset\cdots\subset\mathcal{Z}_{irr}^{\mathcal{X},\infty} \tag{11}$$

and their images by a section of ζ (see Example 2 below)

$$\mathcal{L}_{irr}^{\mathcal{X},\le 2}\subset\cdots\subset\mathcal{L}_{irr}^{\mathcal{X},\le p}\subset\cdots\subset\mathcal{L}_{irr}^{\mathcal{X},\infty}, \tag{12}$$

such that the following restriction is an isomorphism of algebras [9, 10]

$$\zeta:\mathbb{Q}[\mathcal{L}_{irr}^{\infty}(\mathcal{X})]\longrightarrow\mathbb{Q}[\mathcal{Z}_{irr}^{\mathcal{X},\infty}]=\mathcal{Z}. \tag{13}$$

Note that one also has

$$\mathcal{L}_{irr}^{\mathcal{X},\infty}=\bigcup_{p\ge 2}\mathcal{L}_{irr}^{\mathcal{X},\le p}\quad\text{and}\quad\mathcal{L}_{irr}^{\mathcal{X},\infty}=\bigcup_{p\ge 2}\mathcal{L}_{irr}^{\mathcal{X},\le p}. \tag{14}$$

Now, let us describe the algorithm **LocalCoordinateIdentification** below which brings aditional results to [3]. It provides the rewriting systems $(\mathbb{Q}1_{X^*}\oplus x_0\mathbb{Q}\langle X\rangle x_1,\mathcal{R}_{irr}^X)$ and $(\mathbb{Q}1_{Y^*}\oplus(Y\setminus\{y_1\})\mathbb{Q}\langle Y\rangle,\mathcal{R}_{irr}^Y)$ which are without critical pairs, noetherian, confluent and precisely contains the above sets (see (11)–(12)) and, on the other hand, the set of homogenous in weight polynomials, belonging to $\mathbb{Q}[\mathcal{L}yn\mathcal{X}\setminus\text{gDIV}]$, which are image by a section of the surjective ζ polymorphism from $\{\zeta(Q_l)=0\}_{l\in\mathcal{L}yn\mathcal{X}\setminus\text{gDIV}}$. It is denoted by $\mathcal{Q}_\mathcal{X}$:

$$\mathcal{Q}_\mathcal{X}=\{Q_l\}_{l\in\mathcal{L}yn\mathcal{X}\setminus\text{gDIV}} \tag{15}$$

and generates the shuffle or quasi-shuffle ideal $\mathcal{R}_\mathcal{X}$ inside $\ker\zeta$ as follows

$$\mathcal{R}_\mathcal{X}:=\text{span}_\mathbb{Q}\mathcal{Q}_\mathcal{X}\subseteq\ker\zeta. \tag{16}$$

For any $p\ge 2$ and $l\in\mathcal{L}yn^p\mathcal{X}:=\{l\in\mathcal{L}yn\mathcal{X}|(l)=p\}$, any nonzero homogenous in weight polynomial (belonging to $\mathcal{Q}_\mathcal{X}$) $Q_l=\Sigma_l-\Upsilon_l$ (resp. $Q_l=S_l-U_l$) is led by Σ_l (resp. S_l) being transcendent over $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\le p}]$ and $\Upsilon_l=Q_l-\Sigma_l$ (resp. $U_l=Q_l-S_l$) is canonically represented in $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\le p}]$. Then let $\Sigma_l\to\Upsilon_l$ and $S_l\to U_l$ be the rewriting rules, respectively, of

$$\mathcal{R}_{irr}^Y:=\{\Sigma_l\to\Upsilon_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}\quad\text{and}\quad\mathcal{R}_{irr}^X:=\{S_l\to U_l\}_{l\in\mathcal{L}ynX\setminus X}. \tag{17}$$

On the other hand, the following assertions are equivalent (see Example 2 below)

1. $Q_l=0$
2. $\Sigma_l\in\mathcal{L}_{irr}^{Y,\le p}$ (resp. $S_l\in\mathcal{L}_{irr}^{X,\le p}$),
3. $\Sigma_l\to\Sigma_l$ (resp. $S_l\to S_l$).

In the other words, the ordering over $\mathcal{L}yn\mathcal{X}$ induces the ordering over $\mathcal{L}_{irr}^{\mathcal{X},\infty},\mathcal{R}_\mathcal{X},\mathcal{R}_{irr}^\mathcal{X}$ and, in the systems $(\mathbb{Q}1_{X^*}\oplus x_0\mathbb{Q}\langle X\rangle x_1,\mathcal{R}_{irr}^X)$ and $(\mathbb{Q}1_{Y^*}\oplus(Y\setminus\{y_1\})\mathbb{Q}\langle Y\rangle,\mathcal{R}_{irr}^Y)$,

1. each irreducible term, in $\mathcal{L}_{irr}^{\mathcal{X},\infty}$, is an element of the algebraic basis $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$ of $(\mathbb{Q}1_{Y^*}\oplus(Y\setminus\{y_1\})\mathbb{Q}\langle Y\rangle,\;\text{⧢}\!\!\!-)$ (resp. $\{\Sigma_l\}_{l\in\mathcal{L}ynX\setminus X}$ of $(\mathbb{Q}1_{X^*}\oplus x_0\mathbb{Q}\langle X\rangle x_1,\text{⧢})$),
2. each rewriting rule, in $\mathcal{R}_{irr}^\mathcal{X}$, admits the left side being transcendent over $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$ and the right side being canonically represented in $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$. The difference of these two sides belongs to the ordered ideal $\mathcal{R}_\mathcal{X}$ of $\mathbb{Q}[\mathcal{L}yn\mathcal{X}\setminus\text{gDIV}]$.

LocalCoordinateIdentification

```
$\mathcal{Z}_{irr}^{\mathcal{X},\infty} := \{\}; \mathcal{L}_{irr}^{\mathcal{X},\infty} := \{\}; \mathcal{R}_{irr}^{\mathcal{X}} := \{\}; \mathcal{Q}_{\mathcal{X}} := \{\}$;
for $p$ ranges in $2,\ldots,\infty$ do
      for $l$ ranges in the totally ordered $\mathcal{L}yn^p\mathcal{X}$ do
        identify $\langle Z_\gamma|\Pi_l\rangle$ in $Z_\gamma = B(y_1)\pi_Y Z_{\sqcup\!\sqcup}$ and $\langle Z_{\sqcup\!\sqcup}|P_l\rangle$ in $Z_{\sqcup\!\sqcup} = B(x_1)^{-1}\pi_X Z_\gamma$;
        by elimination, obtain equations on $\{\zeta(\Sigma_{l'})\}_{\substack{l'\in\mathcal{L}yn^pY\\ l'\preceq l}}$ and on $\{\zeta(S_{l'})\}_{\substack{l'\in\mathcal{L}yn^pX\\ l'\preceq l}}$ ;
        express[3] the equations led by $\zeta(\Sigma_l)$ and by $\zeta(S_l)$ as rewriting rules;
        if $\zeta(\Sigma_l) \to \zeta(\Sigma_l)$ then $\mathcal{Z}_{irr}^{Y,\infty} := \mathcal{Z}_{irr}^{Y,\infty} \cup \{\zeta(\Sigma_l)\}$ and $\mathcal{L}_{irr}^{Y,\infty} := \mathcal{L}_{irr}^{Y,\infty} \cup \{\Sigma_l\}$
          else $\mathcal{R}_{irr}^{Y} := \mathcal{R}_{irr}^{Y} \cup \{\Sigma_l \to \Upsilon_l\}$ and $\mathcal{Q}_Y := \mathcal{Q}_Y \cup \{\Sigma_l - \Upsilon_l\}$;
        if $\zeta(S_l) \to \zeta(S_l)$ then $\mathcal{Z}_{irr}^{X,\infty} := \mathcal{Z}_{irr}^{X,\infty} \cup \{\zeta(S_l)\}$ and $\mathcal{L}_{irr}^{X,\infty} := \mathcal{L}_{irr}^{X,\infty} \cup \{S_l\}$
          else $\mathcal{R}_{irr}^{X} := \mathcal{R}_{irr}^{X} \cup \{S_l \to U_l\}$ and $\mathcal{Q}_X := \mathcal{Q}_X \cup \{S_l - U_l\}$
      end_for
end_for
```

With the notations introduced in (11)–(17), on also has[4]

Proposition 1 ([9, 10]). 1. *$\mathcal{R}_{\mathcal{X}} = \ker\zeta$ and $\mathbb{Q}[\mathcal{Z}_{irr}^{\mathcal{X},\infty}] = \mathcal{Z} = \operatorname{Im}\zeta$.*

2. *$\mathbb{Q}[\{S_l\}_{l\in\mathcal{L}ynX\setminus X}] = \mathcal{R}_X \oplus \mathbb{Q}[\mathcal{L}_{irr}^{X,\infty}]$ and $\mathbb{Q}[\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}] = \mathcal{R}_Y \oplus \mathbb{Q}[\mathcal{L}_{irr}^{Y,\infty}]$.*

Proof –

1. Let $Q \in \ker\zeta, \langle Q|1_{\mathcal{X}^*}\rangle = 0$. Then $Q = Q_1 + Q_2$ (with $Q_2 \in \mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$ and $Q_1 \in \mathcal{R}_{\mathcal{X}}$). Hence, decomposing in $\{S_l\}_{l\in\mathcal{L}ynX\setminus X}$ (resp. $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$) and reducing by $\mathcal{R}_{irr}^{\mathcal{X}}$, it follows that $Q \equiv_{\mathcal{R}_{irr}^{\mathcal{X}}} Q_1 \in \mathcal{R}_{\mathcal{X}}$ and then the expected result.
   Let $w \in$ CONV. Decomposing in $\{S_l\}_{l\in\mathcal{L}ynX\setminus X}$ (resp. $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$) and reducing by $\mathcal{R}_{irr}^{\mathcal{X}}$, $w \in \mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$. Applying (13) and (5)–(6), $\zeta(w) \in \mathbb{Q}[\mathcal{Z}_{irr}^{\mathcal{X},\infty}] = \mathcal{Z}$ and $\mathcal{Z} = \operatorname{Im}\zeta$. Extending by linearrity, it follows the expected result.
2. For any $w \in$ CONV, decomposing in $\{S_l\}_{l\in\mathcal{L}ynX\setminus X}$ (resp. $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$) and reducing by $\mathcal{R}_{irr}^{\mathcal{X}}$, $\zeta(w) \in \mathbb{Q}[\mathcal{Z}_{irr}^{\mathcal{X},\infty}]$. By linearity, if $P \in \mathbb{Q}[\{S_l\}_{l\in\mathcal{L}ynX\setminus X}]$ (resp. $\mathbb{Q}[\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}]$) and $P \notin \ker\zeta \supseteq \mathcal{R}_{\mathcal{X}}$ then $\zeta(P) \in \mathbb{Q}[\mathcal{Z}_{irr}^{\mathcal{X},\infty}]$.
   On the other hand, if $Q \in \mathcal{R}_{\mathcal{X}} \cap \mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$ then, by (16), $\zeta(Q) = 0$ and then, by (13), $Q = 0$ yielding the expected result.

□

Theorem 1 ([9, 10]). *The $\mathbb{Q}$-algebra $\mathcal{Z}$ is freely generated by $\mathcal{Z}_{irr}^{\mathcal{X},\infty}$ and $\mathcal{Z} = \mathbb{Q}1 \oplus \bigoplus_{k\geq 2}\mathcal{Z}_k$.*

Proof – By (13) and Proposition 1, $\mathcal{Z}$ is freely generated by $\mathcal{Z}_{irr}^{\mathcal{X},\infty}$ and $\ker\zeta$, being generated by the homogenous in weight polynomials $\{Q_l\}_{l\in\mathcal{L}yn\mathcal{X}\setminus\mathrm{gDIV}}$, is graded. With the notations in Conjecture 1, being isomorphic to $\mathbb{Q}1_{Y^*} \oplus (Y\setminus\{y_1\})\mathbb{Q}\langle Y\rangle/\ker\zeta$ and to $\mathbb{Q}1_{X^*} \oplus x_0\mathbb{Q}\langle X\rangle x_1/\ker\zeta$, $\mathcal{Z}$ is also graded. □

Corollary 1 ([9, 10]). *Let $P \in \mathcal{L}_{irr}^{\mathcal{X},\infty}$. Then $\zeta(P)$ is a transcendent number.*

Proof – Let $P \in \mathbb{Q}\langle\mathcal{X}\rangle$ and $P \notin \ker\zeta$, being homogenous in weight, or $P \in$ CONV. Since $\mathcal{Z}_k\mathcal{Z}_{k'} \subset \mathcal{Z}_{k+k'}$ $(k,k' \geq 1)$ then each monomial $(\zeta(P))^k$ $(k \geq 1)$ is of different weight and then, by Theorem 1, $\zeta(P)$ could not satisfy, over $\mathbb{Q}$, an algebraic equation $T^k + a_{k-1}T^{k-1} + \ldots = 0$ meaning that $\zeta(P)$ is a transcendent number. Since any $P \in \mathcal{L}_{irr}^{\mathcal{X},\infty}$ is homogenous in weight then it follows the expected result. □

[3]This step and the following ones are not yet been achieved by the implementation in [3].
[4]See also [11] for further information.

Example 1 (irreducible polyzetas, [3]).

$$
\begin{aligned}
\mathcal{Z}_{irr}^{X,\leq 12} &= \{\zeta(S_{x_0x_1}),\zeta(S_{x_0^2x_1}),\zeta(S_{x_0^4x_1}),\zeta(S_{x_0^6x_1}),\zeta(S_{x_0x_1^2x_0x_1^4}),\zeta(S_{x_0^8x_1}),\\
&\quad \zeta(S_{x_0x_1^2x_0x_1^6}),\zeta(S_{x_0^{10}x_1}),\zeta(S_{x_0x_1^3x_0x_1^7}),\zeta(S_{x_0x_1^2x_0x_1^8}),\zeta(S_{x_0x_1^4x_0x_1^6})\}.\\
\mathcal{Z}_{irr}^{Y,\leq 12} &= \{\zeta(\Sigma_{y_2}),\zeta(\Sigma_{y_3}),\zeta(\Sigma_{y_5}),\zeta(\Sigma_{y_7}),\zeta(\Sigma_{y_3y_1^5}),\zeta(\Sigma_{y_9}),\zeta(\Sigma_{y_3y_1^7}),\\
&\quad \zeta(\Sigma_{y_{11}}),\zeta(\Sigma_{y_2y_1^9}),\zeta(\Sigma_{y_3y_1^9}),\zeta(\Sigma_{y_2^2y_1^8})\}.
\end{aligned}
$$

Example 2 (Rewriting on $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$ and $\{S_l\}_{l\in\mathcal{L}ynX\setminus X}$, irreducible terms).

	Rewriting on $\{\Sigma_l\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$	*Rewriting on* $\{S_l\}_{\mathcal{L}ynX\setminus X}$
3	$\Sigma_{y_2y_1} \to \frac{3}{2}\Sigma_{y_3}$	$S_{x_0x_1^2} \to S_{x_0^2x_1}$
4	$\Sigma_{y_4} \to \frac{2}{5}\Sigma_{y_2}^{\uplus 2}$	$S_{x_0^3x_1} \to \frac{2}{5}S_{x_0x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_3y_1} \to \frac{3}{10}\Sigma_{y_2}^{\uplus 2}$	$S_{x_0^2x_1^2} \to \frac{1}{10}S_{x_0x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_2y_1^2} \to \frac{2}{3}\Sigma_{y_2}^{\uplus 2}$	$S_{x_0x_1^3} \to \frac{2}{5}S_{x_0x_1}^{\sqcup\!\sqcup 2}$
5	$\Sigma_{y_3y_2} \to 3\Sigma_{y_3}\Sigma_{y_2} - 5\Sigma_{y_5}$	$S_{x_0^3x_1^2} \to -S_{x_0^2x_1}S_{x_0x_1} + 2S_{x_0^4x_1}$
	$\Sigma_{y_4y_1} \to -\Sigma_{y_3}\Sigma_{y_2} + \frac{5}{2}\Sigma_{y_5}$	$S_{x_0^2x_1x_0x_1} \to -\frac{3}{2}S_{x_0^4x_1} + S_{x_0^2x_1}S_{x_0x_1}$
	$\Sigma_{y_2^2y_1} \to \frac{3}{2}\Sigma_{y_3}\Sigma_{y_2} - \frac{25}{12}\Sigma_{y_5}$	$S_{x_0^2x_1^3} \to -S_{x_0^2x_1}S_{x_0x_1} + 2S_{x_0^4x_1}$
	$\Sigma_{y_3y_1^2} \to \frac{5}{12}\Sigma_{y_5}$	$S_{x_0x_1x_0x_1^2} \to \frac{1}{2}S_{x_0^4x_1}$
	$\Sigma_{y_2y_1^3} \to \frac{1}{4}\Sigma_{y_3}\Sigma_{y_2} + \frac{5}{4}\Sigma_{y_5}$	$S_{x_0x_1^4} \to S_{x_0^4x_1}$
6	$\Sigma_{y_6} \to \frac{8}{35}\Sigma_{y_2}^{\uplus 3}$	$S_{x_0^5x_1} \to \frac{8}{35}S_{x_0x_1}^{\sqcup\!\sqcup 3}$
	$\Sigma_{y_4y_2} \to \Sigma_{y_3}^{\uplus 2} - \frac{4}{21}\Sigma_{y_2}^{\uplus 3}$	$S_{x_0^4x_1^2} \to \frac{6}{35}S_{x_0x_1}^{\sqcup\!\sqcup 3} - \frac{1}{2}S_{x_0^2x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_5y_1} \to \frac{2}{7}\Sigma_{y_2}^{\uplus 3} - \frac{1}{2}\Sigma_{y_3}^{\uplus 2}$	$S_{x_0^3x_1x_0x_1} \to \frac{4}{105}S_{x_0x_1}^{\sqcup\!\sqcup 3}$
	$\Sigma_{y_3y_1y_2} \to -\frac{17}{30}\Sigma_{y_2}^{\uplus 3} + \frac{9}{4}\Sigma_{y_3}^{\uplus 2}$	$S_{x_0^3x_1^3} \to \frac{23}{70}S_{x_0x_1}^{\sqcup\!\sqcup 3} - S_{x_0^2x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_3y_2y_1} \to 3\Sigma_{y_3}^{\uplus 2} - \frac{9}{10}\Sigma_{y_2}^{\uplus 3}$	$S_{x_0^2x_1x_0x_1^2} \to \frac{2}{105}S_{x_0x_1}^{\sqcup\!\sqcup 3}$
	$\Sigma_{y_4y_1^2} \to \frac{3}{10}\Sigma_{y_2}^{\uplus 3} - \frac{3}{4}\Sigma_{y_3}^{\uplus 2}$	$S_{x_0^2x_1^2x_0x_1} \to -\frac{89}{210}S_{x_0x_1}^{\sqcup\!\sqcup 3} + \frac{3}{2}S_{x_0^2x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_2^2y_1^2} \to \frac{11}{63}\Sigma_{y_2}^{\uplus 3} - \frac{1}{4}\Sigma_{y_3}^{\uplus 2}$	$S_{x_0^2x_1^4} \to \frac{6}{35}S_{x_0x_1}^{\sqcup\!\sqcup 3} - \frac{1}{2}S_{x_0^2x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_3y_1^3} \to \frac{1}{21}\Sigma_{y_2}^{\uplus 3}$	$S_{x_0x_1x_0x_1^3} \to \frac{8}{21}S_{x_0x_1}^{\sqcup\!\sqcup 3} - S_{x_0^2x_1}^{\sqcup\!\sqcup 2}$
	$\Sigma_{y_2y_1^4} \to \frac{17}{50}\Sigma_{y_2}^{\uplus 3} + \frac{3}{16}\Sigma_{y_3}^{\uplus 2}$	$S_{x_0x_1^5} \to \frac{8}{35}S_{x_0x_1}^{\sqcup\!\sqcup 3}$

$$
\begin{aligned}
\mathcal{L}_{irr}^{X,\leq 12} &= \{S_{x_0x_1},S_{x_0^2x_1},S_{x_0^4x_1},S_{x_0^6x_1},S_{x_0x_1^2x_0x_1^4},S_{x_0^8x_1},\\
&\quad S_{x_0x_1^2x_0x_1^6},S_{x_0^{10}x_1},S_{x_0x_1^3x_0x_1^7},S_{x_0x_1^2x_0x_1^8},S_{x_0x_1^4x_0x_1^6}\}.\\
\mathcal{L}_{irr}^{Y,\leq 12} &= \{\Sigma_{y_2},\Sigma_{y_3},\Sigma_{y_5},\Sigma_{y_7},\Sigma_{y_3y_1^5},\Sigma_{y_9},\Sigma_{y_3y_1^7},\Sigma_{y_{11}},\Sigma_{y_2y_1^9},\Sigma_{y_3y_1^9},\Sigma_{y_2^2y_1^8}\}.
\end{aligned}
$$

3. Conclusion

Thanks to a Abel like theorem and the equation bridging the algebraic structures of the $\mathbb{Q}$-algebra $\mathcal{Z}$ generated by the polyzetas [5], the algorithm **LocalCoordinateIdentification** provides the algebraic relations[5] among the local coordinates, of second kind on the groups of group-like series, of the noncommutative series $Z_{\sqcup\!\sqcup}$ (*i.e.* $\{\zeta(S_l)\}_{l\in\mathcal{L}ynX\setminus X}$) and $Z_{\uplus}$ (*i.e.* $\{\zeta(\Sigma_l)\}_{l\in\mathcal{L}ynY\setminus\{y_1\}}$). These relations constitute two confluent rewriting systems in which the irreducible terms, belonging to $\mathcal{Z}_{irr}^{\mathcal{X},\infty}$, represent the algebraic generators for $\mathcal{Z}$ and, on the other hand, the $\sqcup\!\sqcup$-ideal $\mathcal{R}_X$ and the $\uplus$-ideal $\mathcal{R}_Y$ represent the kernels of the ζ polymorphism (Proposition 1). These ideals are generated by the polynomials, totally ordered and homogenous in weight, $\{Q_l\}_{l\in\mathcal{L}yn\mathcal{X}\setminus\mathrm{gDIV}}$ and are interpreted as the confluent rewriting systems in which the irreducible terms belong to $\mathcal{L}_{irr}^{\mathcal{X},\infty}$ and, in each rewriting rule of $\mathcal{R}_{irr}^{\mathcal{X}}$, the left side

[5]These are different from those among $\{\zeta(l)\}_{l\in\mathcal{L}yn\mathcal{X}\setminus\mathrm{gDIV}}$ obtained by "double shuffle relations" [8], for which Conjecture 1 holds, up to weight 10.

is the leading monomial of $Q_l, l \in \mathcal{L}yn\mathcal{X} \setminus$ gDIV and is transcendent over $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$ while the right side is canonically represented on $\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}]$. It follows that $\zeta(\mathbb{Q}[\mathcal{L}_{irr}^{\mathcal{X},\infty}])$, *i.e.* $\mathcal{Z}$, as being isomorphic to $\mathbb{Q}1_{X^*} \oplus x_0\mathbb{Q}\langle X\rangle x_1/\mathcal{R}_X$ and to $\mathbb{Q}1_{Y^*} \oplus (Y \setminus \{y_1\})\mathbb{Q}\langle Y\rangle/\mathcal{R}_Y$, is $\mathbb{Q}$-free and graded (Theorem 1) and then irreducible polyzetas, being $\mathbb{Q}$-algebraic independent, are transcendent numbers (Corollary 1). By these results, up to weight 12, Conjecture 1 holds (see also[6] [7, 12]), *i.e.* $\mathcal{Z}_{irr}^{\mathcal{X},\leq 12}$ is $\mathbb{Q}$-algebraically free (Example 2).

References

[1] J. Blümlein, D. J. Broadhurst, and J. A. M. Vermaseren.– *The multiple zeta value data mine*, *Computer Physics Communications*, 181(3), pp. 582-625, 2010.

[2] D. Borwein, J.M. Borwein & R. Girgensohn.– *Explicit evaluation of Euler sums*. Proc. Edin. Math. Soc., 38 (1995), pp. 277-294.

[3] V.C. Bui, G.H.E. Duchamp, V. Hoang Ngoc Minh.– *Structure of Polyzetas and Explicit Representation on Transcendence Bases of Shuffle and Stuffle Algebras*, J. Sym. Comp. 83 (2017).

[4] V.C. Bui, V. Hoang Ngoc Minh, Q.H. Ngo and V. Nguyen Dinh.– *On The Kernels Of The Zeta Polymorphism*, to appear in the proceeding of “XV International Workshop Lie Theory and Its Applications in Physics".

[5] C. Costermans, V. Hoang Ngoc Minh.– *Noncommutative algebra, multiple harmonic sums and applications in discrete probability*, J. Sym. Comp. (2009), 801-817.

[6] L. Euler.– *Meditationes circa singulare serierum genus*, Novi. Comm. Acad. Sci. Petropolitanae, 20 (1775), 140-186.

[7] M. Espie, J.-C. Novelli, G. Racinet.– *Formal Computations About Multiple Zeta Values*, IRMA Lect. Math. Theor. Phys. 3, de Gruyter, Berlin, 2003, pp. 1-16.

[8] V. Hoang Ngoc Minh, M. Petitot.– *A Lyndon words, polylogarithms and the Riemann ζ function*, Proceedings of FPSAC’98, 1998.

[9] V. Hoang Ngoc Minh.– *Structure of polyzetas and Lyndon words*, Vietnamese Math. J. (2013), 41, Issue 4, 409-450.

[10] V. Hoang Ngoc Minh.– *On the solutions of universal differential equation with three singularities*, in Confluentes Mathematici, Tome 11 (2019) no. 2, p. 25-64.

[11] V. Hoang Ngoc Minh.– *On the Algebraic Bases of Polyzetas*, in submission.

[12] M. Kaneko, M. Noro, and K. Tsurumaki.– *On a conjecture for the dimension of the space of the multiple zeta values*, IMA Volumes in Mathematics and its Applications 148, Springer, New York, 2008, pp. 47-58.

[13] D. Zagier.– *Values of zeta functions and their applications*, in “First European Congress of Mathematics", vol. 2, Birkhäuser, pp. 497-512, (1994).

[6]All these implementations base on the “double shuffle relations" and provide linear relations.

An e-origami artwork of a big wing crane

Tetsuo Ida[†]

University of Tsukuba, 1-1-1 Tennoudai, Tsukuba, 305-8573, Japan

Abstract
We present the construction of a big wing crane as an application of folding with virtual cutting and gluing edges of origami faces in the e-origami environment. With the new methodology, we can reason about the construction algorithm for classical origami in finer steps rather than relying on the skill of the human hand. We employ a new origami model, as described in our earlier work, and we have developed software that implements classical origami folds.

Keywords
e-origami, classical fold, paper folding rule, cut and glue edges

1. Introduction

In our previous paper [1], we introduced a new technique of *cut-and-glue of a shared face edge* to e-origami[1]. We observed that an origami artwork is a complex arrangement of bounded two-sided flat planes, or faces, intricately connected and superposed by repeated folds of a single (virtual) sheet of paper. This collection of faces culminates in an object with a remarkable shape.

Our research demonstrates that cutting an edge shared by two faces unveils a class of classical folds. By gluing the faces divided by the cut, we restore the connection of the separated faces. This cut-and-glue technique opens up vast possibilities, enabling the discovery of new folds that were previously deemed impossible by Huzita-Justin folds, which, when applied to practical constructions, have certain limitations that our approach overcomes [2] and [3]. The inside reverse fold, one of the most straightforward classical folds, is not included in the Huzita-Justin folds. When we apply the cut-and-glue technique, we can realize the inside reverse fold by combining Huzita-Justin folds. We demonstrate the practical application of our method by constructing a big wing crane, a well-versed sophisticated origami structure [4] that demands a deep research investigation.

2. Modeling for e-origami

Origami is a term meaning “folding paper.” It also refers to a sheet of paper used for origami. Folding an origami along a fold line and unfolding the fold to the previous shape leaves a line segment, called a *crease*, on the origami. We can construct various interesting geometric objects when we freely choose fold lines and allow overlaps of faces without breaking the original sheet. When we impose mathematically plausible fold line construction rules, we can define an origami geometry that deserves deep mathematical investigation.

Euclidean (plane) geometry constructs geometrical objects using only a straightedge and a compass. Similarly, origami geometry, a tool-less approach, defines its rules. Huzita-Justin’s rules are the commonly agreed rule set on which origami geometry is based. In Table 1, we list some (4 out of 7) of the Huzita-Justin rules implemented in the Eos system [5] that we use to construct a big wing crane. These rules, together with newly implemented classical folds (see next section) and software tools of Eos, allow us to manipulate origami flexibly.

SCSS 2024: The 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
ida@cs.tsukuba.ac.jp (T. Ida)
https://www.i-eos.org/ida (T. Ida)
0000-0002-5683-216X (T. Ida)

[1]We use the term *e-origami* to refer to origami innovated by information technology.

Table 1
Huzita-Justin rules

Rule	Command	Operation
(O1)	HO$[PQ]$	Fold along line PQ
(O2)	HO$[P, G]$	Fold to superpose point P and point Q
(O3)	HO$[PQ, RS]$	Fold to superpose line PQ and line RS
(O4)	HO$[PQ, X]$	Fold along the line perpendicular to line PQ that passes through point X

3. Modeling for e-origami

We give a mathematical notation for the definition of origami and reason about its properties. In origami, we have faces and two kinds of neighborhood relations between the faces, i.e., superposition and adjacency. The superposition is a vertical neighborhood relation, and the adjacency is a horizontal one. A face is a polygon having an attribute of sides. When we denote polygon $P_1 \dots P_n$, where we arrange points $P_1 \dots P_n$ counterclockwise, we call the plane *front* side, and the polygon $P_n \dots P_1$ is identified the same as with polygon $P_1 \dots P_n$ with the attribute of the *back* side. For any faces, $f = P_1 \dots P_n$ and $g = Q_1 \dots Q_n$, f is equal to g iff g is a cyclic permutation of f or a cyclic permutation of a reverse of f.

We let Π be a finite set of faces, $\backsim$ be a binary relation on Π, called *adjacency relation*, and $\succ$ be a binary relation on Π, called *superposition relation.* An abstract origami is a structure $(\Pi, \backsim, \succ)$. We abbreviate abstract origami to *AO*. We denote the set of AOs by $\mathbf{O}$. An abstract origami system is an abstract rewriting system $(\mathbf{O}, \looparrowright)$ [6] , where $\looparrowright$ is a rewrite relation on $\mathbf{O}$, called *abstract fold.*

For $\mathcal{O}, \mathcal{O}'(\in \mathbf{O})$, we write $\mathcal{O} \looparrowright \mathcal{O}'$ when $\mathcal{O}$ is abstractly folded to $\mathcal{O}'$. We begin an origami construction with an initial AO and perform an abstract fold repeatedly until we obtain the desired AO. Usually, we start an origami construction with a square sheet of paper. This initial sheet of paper is abstracted as a structure having a single distinguished face denoted by the numeral 1. Then, the initial AO $\mathcal{O}_1$ is represented by $(\{1\}, \emptyset, \emptyset)$. Furthermore, when we fold face n, the face is divided into two faces $2n$ and $2n + 1$. We use this convention in this paper and the realization of the data structure of the e-origami system Eos [7] and the origami language Orikoto of Eos. Suppose that we are at the beginning of step i of the construction, having AO $\mathcal{O}_{i-1} = (\Pi_{i-1}, \backsim_{i-1}, \succ_{i-1})$. We perform an abstract fold and obtain a next AO $\mathcal{O}_i = (\Pi_i, \backsim_i, \succ_i)$. Thus, we have the following $\looparrowright$-sequence.

$$\mathcal{O}_1 \looparrowright \mathcal{O}_2 \looparrowright \cdots \looparrowright \mathcal{O}_n$$

An abstract origami construction is a finite $\looparrowright$-sequence of AOs. In concrete terms, the operation $\looparrowright$ can be a fold by one of Huzita-Justin's rules, a mountain fold, a valley fold, etc., each requiring arguments of different kinds. We abuse $\mathcal{O}_i$ to be a name of the origami constructed at step i.

We now move on to concrete origami. Origami is a term meaning "folding paper." It also refers to a sheet of paper used for origami. Folding an origami along a fold line and unfolding the fold to the previous shape leaves a line segment, called a *crease*, on the origami. We can construct various interesting geometric objects when we freely choose fold lines and allow overlaps of faces without breaking the original sheet. When we impose mathematically plausible rules for choosing fold lines, we can define an origami geometry that deserves deep mathematical investigation.

Euclidean (plane) geometry constructs geometrical objects using only a straightedge and a compass. Similarly, origami geometry, a tool-less approach, defines its rules. Huzita-Justin's rules are the commonly agreed rule set on which origami geometry is based. In Table 1, we list some (4 out of 7) Huzita-Justin rules that we use to construct a big wing crane. These rules, newly implemented classical folds (to be discussed in the next section), and our software tools allow us to manipulate origami flexibly.

4. Classical Folds

4.1. Mountain fold and Valley fold

Understanding the fundamental fold operations of the mountain and valley fold is crucial in origami. These folds are named after the resemblance of the crease to a mountain ridge and a valley lap. The crease is formed by the unfold operation that follows the mountain (or valley) fold. The result of the valley fold is shown in Fig.1(b). The origami is now two-layered, but the back layer is not visible since the upper triangle face completely overlaps the lower triangle face. Next, we unfold the origami shown in Fig. 1(c). Unfold does not mean "undoing," although we recover the shape of the origami to the one in Fig.1(a) except for the dotted line segment CA. We call the line segment valley crease. If we have a rich imagination, the crease looks like a lap in the valley.

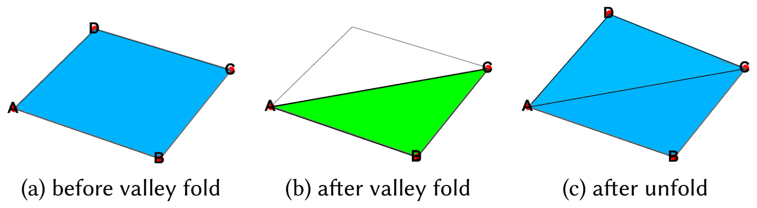


Figure 1: Valley fold

Similarly, we have a mountain fold below.

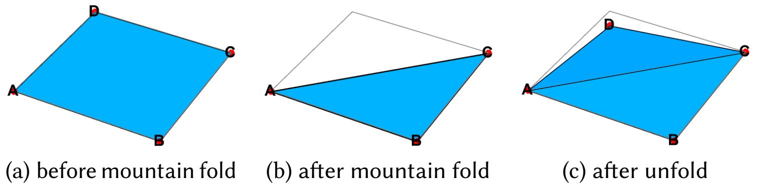


Figure 2: Mountain fold

The mountain and valley folds are similar to the Huzita-Justin rule (O1), with the important difference that rule (O1) operates on all the stacked faces. In contrast, the mountain and valley folds apply to automatically selected faces.

The terms "valley" and "mountain" are sometimes misleading to beginner origami hobbyists since those folds are not necessarily immediately followed by an unfold. Hence, valley and mountain creases may not appear unless the origami is entirely unfolded. More geometrically clear terminology is desired for origami geometers.

4.2. FO

The above observation led us to define command FO. FO, standing for Fold Origami, is a command FO[θ][*faces*, *ray*] for rotating target faces along *ray* by angle θ. It is a generalization of the mountain and valley folds. The implementation of FO contains an algorithm to select the target faces to be rotated based on *faces*.

Note that FO is defined as a Curried function. Using FO, we could define ValleyFold and MountainFold as follows:

```
ValleyFold = FO[-π]; MountainFold = FO[π];
```

With θ other than $\pm\pi$, FO constructs a 3D origami in general. Usually, we use FO[θ] towards the ending steps of the construction.

4.3. Inside reverse and outside reverse folds

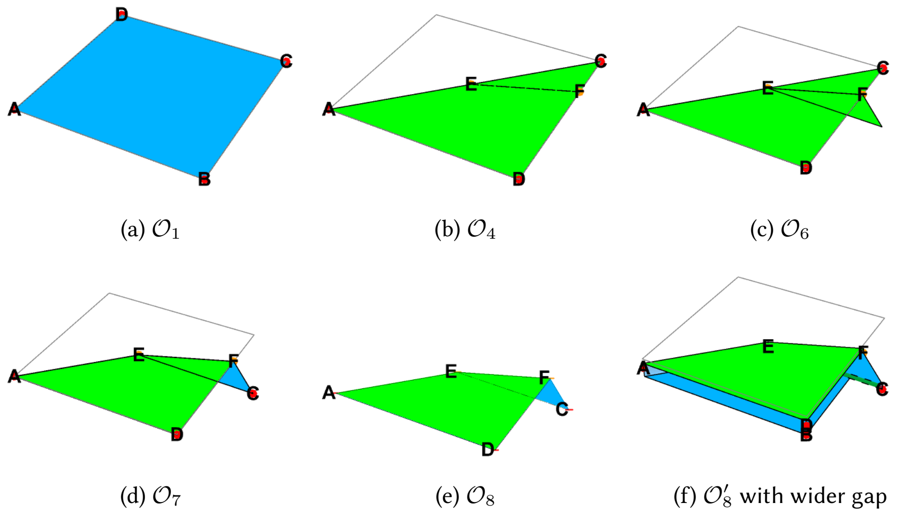
The inside reverse and outside reverse folds are often used in paper folding. Both work on a pair of superposed faces that share an edge. When the cut-and-glue technique is introduced, both folds are realized by combinations of the valley and mountain folds. However, each uses the valley and mountain folds on opposite faces.

4.3.1. Inside reverse fold

Below we show a simple example of an inside reverse fold. Let

$$\mathcal{O}_1 \looparrowright^* \mathcal{O}_4 \looparrowright^* \mathcal{O}_6 \looparrowright \mathcal{O}_7 \looparrowright \mathcal{O}_8$$

be a construction sequence of the example. Each origami $\mathcal{O}_i, i \in \{1, 4, 6, 7, 8\}$ is visualized in Fig. 3.



(a) $\mathcal{O}_1$ (b) $\mathcal{O}_4$ (c) $\mathcal{O}_6$

(d) $\mathcal{O}_7$ (e) $\mathcal{O}_8$ (f) $\mathcal{O}'_8$ with wider gap

Figure 3: Inside reverse fold

Origami $\mathcal{O}_4$ is consructed from $\mathcal{O}_1$ by a sequence of commands; VallyFold, ValleyFold, and Unfold. It is a double-layered stack of faces. On the top layer are two faces, i.e., face CFE and face AEFD. On the second one, i.e., the bottom layer, are two faces of the same shapes as the ones above but in opposite orientations. Each face is identified by a unique face ID, automatically assigned by the system. In this elementary example, we do not have to be concerned with face IDs since it is unnecessary to specify to which faces we apply the inside reverse fold. To the example $\mathcal{O}_4$ of Fig. 3(b) we apply the following command:

```
InsideReverseFold["CE", "FE"]
```

The first argument, CE, specifies the edge to be cut. It is specified as the type ray. The second argument, FE, is the ray along which mountain and valley folds are performed. The above inside reverse fold command performs the following operations:

1. Check if the inside reverse fold is feasible. Namely, check if $\angle\text{CFE} \leq \pi/2$.
2. Cut the edge CE. As a result, point C is split into C and C_1(not shown). The result is $\mathcal{O}_5$. Origami $\mathcal{O}_5$ is not shown in Fig. 3.
3. Valley fold along ray FE. The face C_1EG is moved. *We impose a rule that faces to the right of a fold line (interpreted as a ray) are moved by a fold.* The result is $\mathcal{O}_6$,
4. Mountain fold along ray FE. The face CFE is moved. The result is $\mathcal{O}_7$.
5. Glue the moved edges CE and C_1E to form a new edge CE. The result is $\mathcal{O}_8$.

Note in passing that performing Steps 2 and 3 above in sequence would only be possible when we cut and separated the shared edge EC. As we construct origami in a virtual space, we can cut CE and glue the moved CEs without complication. After following the above steps, the obtained origami is shown in Fig. 3(e) and (f); the latter illustrating the ins and outs of the origami object. Our specially designed viewer generates this graphics image, providing a comprehensive view of the origami.

On the other hand, in the case of the outside reverse fold, we make a polygonal cover on the faces.

4.3.2. Outside reverse fold

An outside reverse fold is similar to an inside reverse fold. The difference is that the outside reverse fold applies mountain and valley folds to different faces in opposite layers. We refer to the construction sequence

$$\mathcal{O}_1 \looparrowright \mathcal{O}_2 \looparrowright \cdots \looparrowright \mathcal{O}_8,$$

where $\mathcal{O}_i, i = 1, \ldots, 8$ are visualized in Fig. 4(a)∼(e). We apply the following command:

```
OutsideReverseFold["CE","FE"]                    (1)
```

to $\mathcal{O}_4$, and obtain $\mathcal{O}_8$. Note that constraint $\pi/2 \leq \angle CEF \leq \pi$ should be satisfied; otherwise the execution of (1) fails. The origami construction sequence in Fig. 4 is now self-explanatory. Figure 4 (f) shows the origami structure of $\mathcal{O}_8$ more clearly.

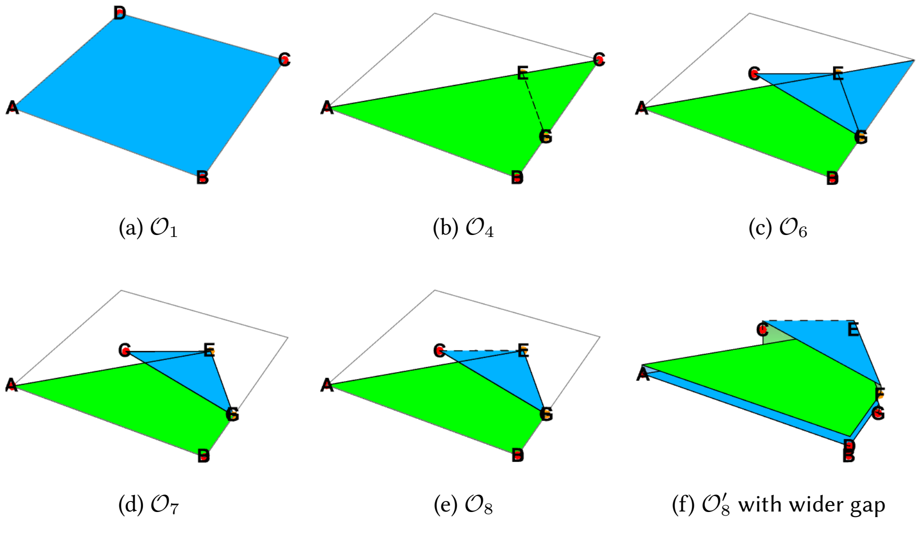


Figure 4: Outside reverse fold

4.3.3. Other classical folds

In addition to the classical folds we have discussed so far, we analyzed the folding algoriths of the following classical folds well-known in the origami community and implemented them: SquashFold, RabbitEarFold, SwivelFold, InsideCrimpPleatFold, and OutsideCrimpPleatFold, using the cut-and-glue technique. They are given in the Appendix.

5. Construction of a big wing crane

5.1. Overview

A crane origami is an exciting example. It is one of the best-known artworks, requiring the classical folds we discussed. Making a delicate crane by hand is challenging for beginner origami hobbyists. In the context of e-origami, this example poses another challenge in modeling a new class of folds. We will present the construction of a big wing crane origami. However, instead of describing in natural language with guiding icons and annotation, we present an algorithm for constructing a crane origami. This algorithm provides a step-by-step prescription for folding the crane in a programming language. We present a big wing crane, a well-known flying crane origami variation [4]. Algorithmically, it is simpler than the ordinary, well-known ones.

We start with a rhombus-shaped piece of paper. We have included entire program codes in a separate webpage (www.i-eos.org/orikoto-program-of-a-big-wing-crane) to help the reader better understand the internal origami structure. These codes are not just for show-they allow us to visualize the intricate folds and their arrangement, making the construction process more accessible. In this construction, we specify the origami faces of textured patterns on the completed work, adding an artistic dimension to the origami. This approach, requiring a special texture mapping algorithm, opens up new creative possibilities in origami modeling. To complete the artwork shown below, we need 71 steps. In one step, an origami object will make one structural change. For example, we count one structural change of the origami object in one application of a valley fold, a mountain fold, and one of the Huzita-Justin folds. One inside reverse fold requires four substeps, i.e., cut, valley fold, mountain fold, and glue. The number of steps for the entire construction is surprisingly large at first sight, but we should observe that the origami objects have several symmetries. Therefore, a similar code sequence is repeated more than once. The number of crucial operations is limited. While the space is limited in this paper, we assure the readers that they grasp the essence of the construction by showing carefully selected crucial steps and output graphics of the origami construction process.

The construction consists of the following four stages.

(1) bird base construction,
(2) leg construction,
(3) bill construction,
(4) wing construction.

We will briefly explore the steps in each stage.

5.2. Bird base construction

LLet τ be a construction $\mathcal{O}_1 \looparrowright^* \mathcal{O}_k$, where $k > 1$ and $\mathcal{O}_k$ has a certain distinguidh feature $\mathcal{A}$, we call τ an $\mathcal{A}$-base construction. We want to construct a bird-like origami $\mathcal{O}_k$. In this stage, we will construct a bird base, i.e., $\mathcal{A}$ is a bird and $k = 49$. We also call $\mathcal{O}_k$ a bird base.

Let τ be a construction $\mathcal{O}_1 \looparrowright^* \mathcal{O}_k$, where $k > 1$ and $\mathcal{O}_k$ has a certain distinguidh feature $\mathcal{A}$, we call τ an $\mathcal{A}$-base construction. We want to construct a bird-like origami $\mathcal{O}_k$. In this stage, we will construct a bird base, i.e., $\mathcal{A}$ is a bird and $k = 49$. We also call $\mathcal{O}_k$ a bird base.

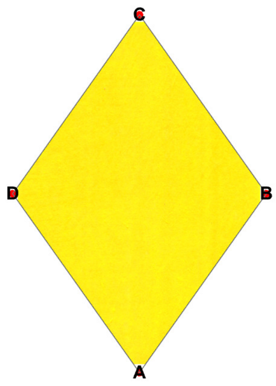
The first 11 steps in τ produce $O_1, \ldots, O_{11}$ shown in Fig. 5. We start with an initial origami of a rhombus shape to make the crane's wings bigger than the crane made from the square initial origami. Applying rule (O2) to $\mathcal{O}_1$ and $\mathcal{O}_2$, we obtain $\mathcal{O}_3$. $\mathcal{O}_3$ is quad layered. We apply rule (O3) to fold $\mathcal{O}_3$ along the bisector of ∠BCE, and then unfold $\mathcal{O}_4$ to obtain $\mathcal{O}5$. Steps 4 and 5 aim to construct a valley crease F4E that will be used in the inside reverse fold on $\mathcal{O}_5$. $\mathcal{O}_5$ has three other creases, F3E, F2E, and F1E, at the intersection of the bisector and the hypotenuses of the right triangular faces on four layers.

The subsequence of the construction $\mathcal{O}_5 \looparrowright^* \mathcal{O}_9 \looparrowright \mathcal{O}_{10}$ shows the first application of the inside reverse fold on $\mathcal{O}_5$:

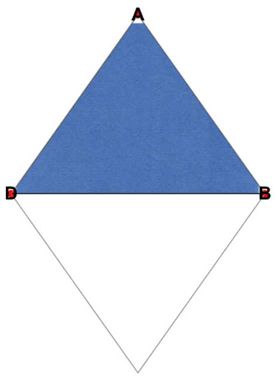
```
InsideReverseFold}["BE", "F4E"].
```

The application of InsideReverseFold requires four substeps, i.e., cut, valley fold, mountain fold, and glue, and returns $\mathcal{O}_9$. The top view of $\mathcal{O}_5$ and $\mathcal{O}_9$ appears the same, but two triangular faces BF3E and BF4E have been moved below face CEF4 in $\mathcal{O}_9$ by the application. When we move face CEF4 at step 10, we observe the difference. By turning over $\mathcal{O}_{10}$, we have origami $\mathcal{O}_{11}$ seen from the backside[2].

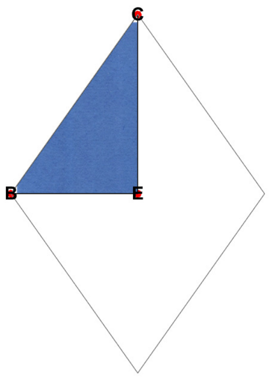
[2]We have a command TurnOver[ray], which is a variation of FO[π][ray].



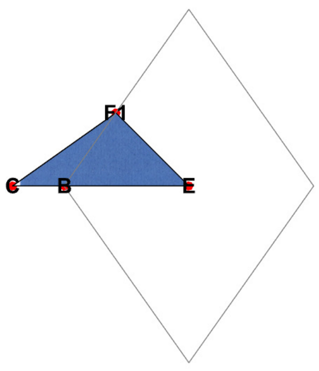
(a) $\mathcal{O}_1$



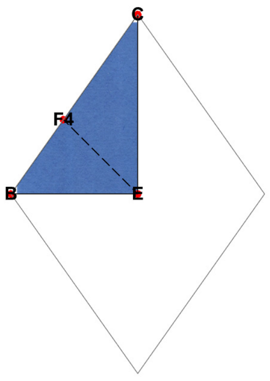
(b) $\mathcal{O}_2$



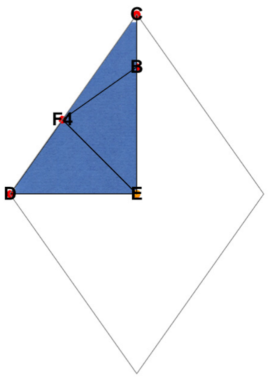
(c) $\mathcal{O}_3$



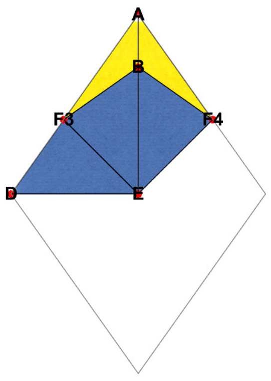
(d) $\mathcal{O}_4$



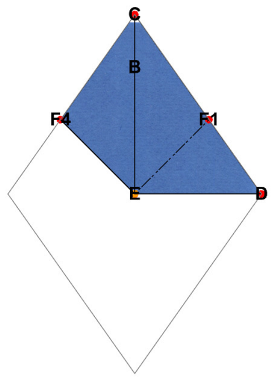
(e) $\mathcal{O}_5$



(f) $\mathcal{O}_9$



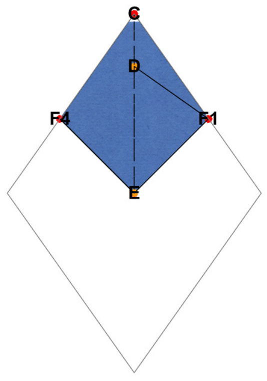
(g) $\mathcal{O}_{10}$



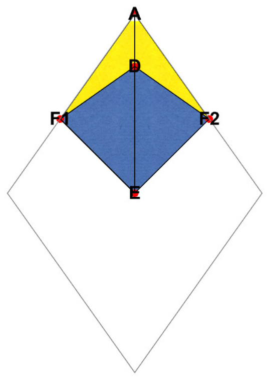
(h) $\mathcal{O}_{11}$

Figure 5: Subsequence 1 of the bitd base

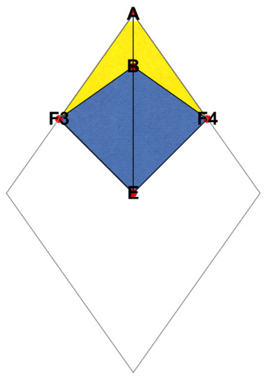
Similarly, we apply another inside reverse fold to the right half of $\mathcal{O}_{11}$, and by following the construction sequence $\mathcal{O}_{11} \looparrowright^* \mathcal{O}_{15} \looparrowright \mathcal{O}_{16} \looparrowright \mathcal{O}_{17}$.



(a) $\mathcal{O}_{15}$: by insert reverse fold



(b) $\mathcal{O}_{16}$: by (O1)



(c) $\mathcal{O}_{17}$: by turn over

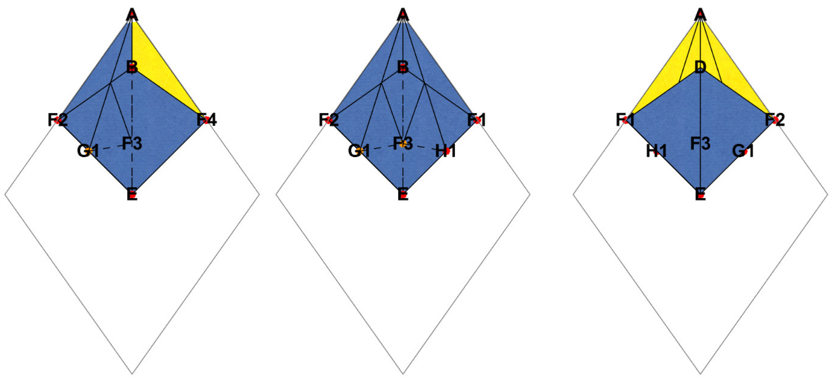
Figure 6: Subsequence 2 of the bird base

The rest of the subsequence in the bird base is given in Figs. 7 and 8 with annotation in the title of each sub-figure. Each sequence of figures are the visualization of

$$\looparrowright^{*} \mathcal{O}_{23} \looparrowright \mathcal{O}_{29} \looparrowright \mathcal{O}_{30},$$

and

$$\looparrowright^{*} \mathcal{O}_{36} \looparrowright \mathcal{O}_{42} \looparrowright \mathcal{O}_{43} \looparrowright^{*} \mathcal{O}_{46} \looparrowright \mathcal{O}_{48} \looparrowright \mathcal{O}_{49}.$$



(a) $\mathcal{O}_{23}$: by insert reverse fold (b) $\mathcal{O}_{29}$: by insert reverse fold (c) $\mathcal{O}_{30}$: by turn over

Figure 7: Subsequence 3 of the bird base

At the end of the bird base construction, we obtain O_{48} and $\mathcal{O}_{49}$. The latter is the bird base. The bird base is crucial in origami crane construction, whether for the big wing crane or for a commonly known classical crane. This slim diamond-shaped origami piece has a crack in the middle of the upper half shown in $\mathcal{O}_{49}$ in Fig. 8(f). This piece is quad-layered, and each double layer is vertically symmetric as well as horizontallly symmetric with respect fo the axis AB.

5.3. Leg construction

The leg construction subsequence is the following:

$$\looparrowright^{*} \mathcal{O}_{53} \looparrowright^{*} \mathcal{O}^{*}_{57} \looparrowright^{*} \mathcal{O}_{61}$$

We apply the inside reverse fold to the right and left parts of the bird base O_{49}. Note that the inside reverse fold can be applied to multiple layers of faces.

5.4. Bill construction

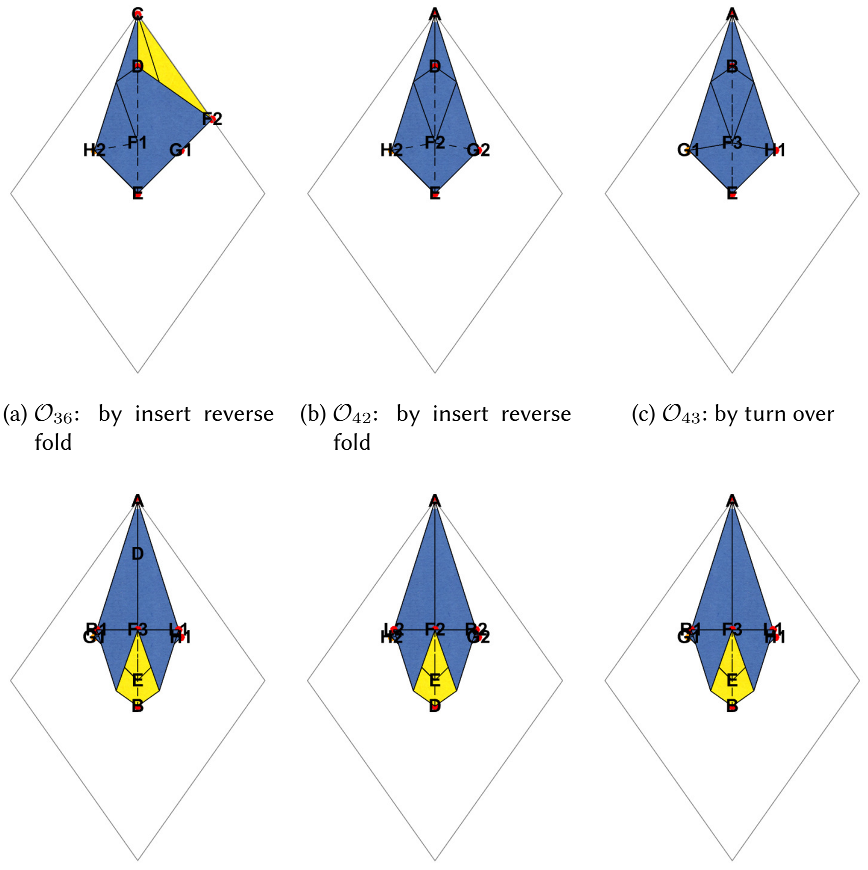
The bill construction subsequence is the following:

$$\looparrowright^{*} \mathcal{O}_{61} \looparrowright^{*} \mathcal{O}_{63} \looparrowright^{*} \mathcal{O}_{67} \looparrowright \mathcal{O}_{68}$$

Here, we make crease Z1Z2, where we choose by the designer's preference arbitraly positions of points Z1 and Z2 on the edges. The crease determines the twist of the bill. We apply the inside reverse fold on O_{62}. Then, we rotate O_{66} along R1L1 by π and obtain O_{68}.

5.5. Wing construction

Finally, we apply the command FO twice to make the origami three-dimensional. We apply FO[-(3/8) π] to those faces that constitute the wings. We complete the construction to bring the bill partly to the right, as this is our preferred posture. Our viewer can manipulate the origami object. The viewer and



(a) $\mathcal{O}_{36}$: by insert reverse fold
(b) $\mathcal{O}_{42}$: by insert reverse fold
(c) $\mathcal{O}_{43}$: by turn over
(d) $\mathcal{O}_{46}$: bring inner face to outside
(e) $\mathcal{O}_{48}$: bring inner face to outside
(f) $\mathcal{O}_{49}$: by turn over

Figure 8: Subsequence 4 of the bird base

Mathematica's graphics functions [8], for example, remove the point names and change the size of the image and lighting.

$$\looparrowright^{*} \mathcal{O}_{69} \looparrowright \mathcal{O}_{70} \looparrowright \mathcal{O}_{71}$$

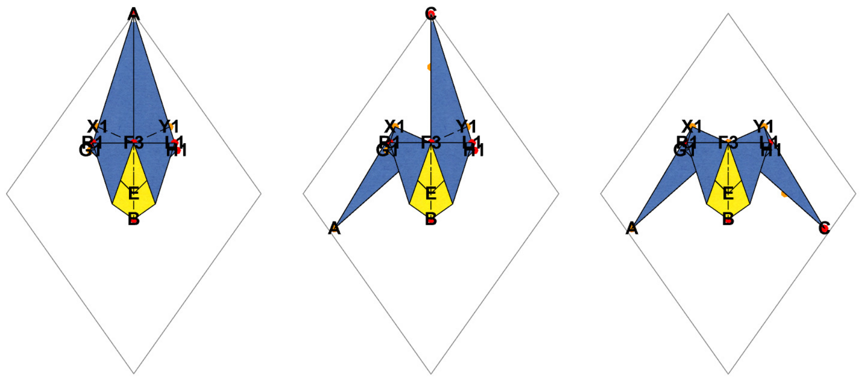
5.6. Texture mapping and adjustment of posture

The following are the results of our artwork. Figure 11(a) is the polished version of $\mathcal{O}_{71}$. We removed the point names, enlarged the image, and readjusted the posture of the crane. Figures 11(b) and (c) are obtained by adding textures to the image of $\mathcal{O}_{71}$, using the functionality of texture mapping of Mathematica.

6. Concluding remarks

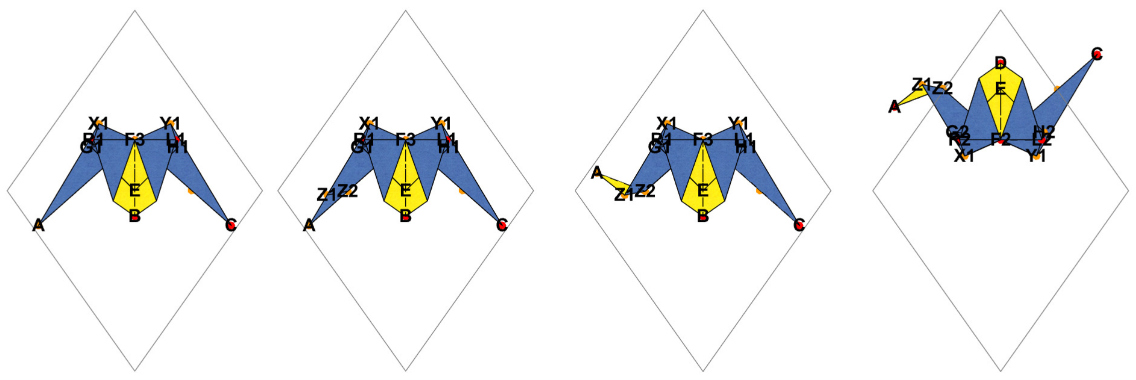
We have shown that the cut-and-glue technique simplifies modeling the classical folds and, hence, the implementation in the e-origami environment. Using a cut operation, we have shown that the seemingly complex inside (and outside) reverse fold is reduced to a sequence of FO folds. We illustrated other popular classical folds, such as a rabbit ear fold, are realized similarly.

We constructed a big wing crane origami as a nontrivial application of the newly defined inside reverse fold. The design is coordinate-free yet allows for the freedom of choice of wing angles and



(a) $\mathcal{O}_{53}$: creae X1F3 (b) $\mathcal{O}_{57}$: by insert reverse fold (c) $\mathcal{O}_{61}$: right leg construction

Figure 9: Leg construction



(a) $\mathcal{O}_{61}$ (b) $\mathcal{O}_{63}$ (c) $\mathcal{O}_{67}$ (d) $\mathcal{O}_{68}$

two positions of bending points of a bill. The construction proceeds as step-by-step fold operations described in the Orikoto origami programming language. Thus, creating a big wing crane is purely an algorithmic process. Furthermore, the added feature of texture mapping of our system, Eos, makes the final product more original artwork.

The algorithm's implementation in the earlier paper [1] was extensively tested and extended for more capabilities. The newer Eos version and the construction program are published at the website [9].

References

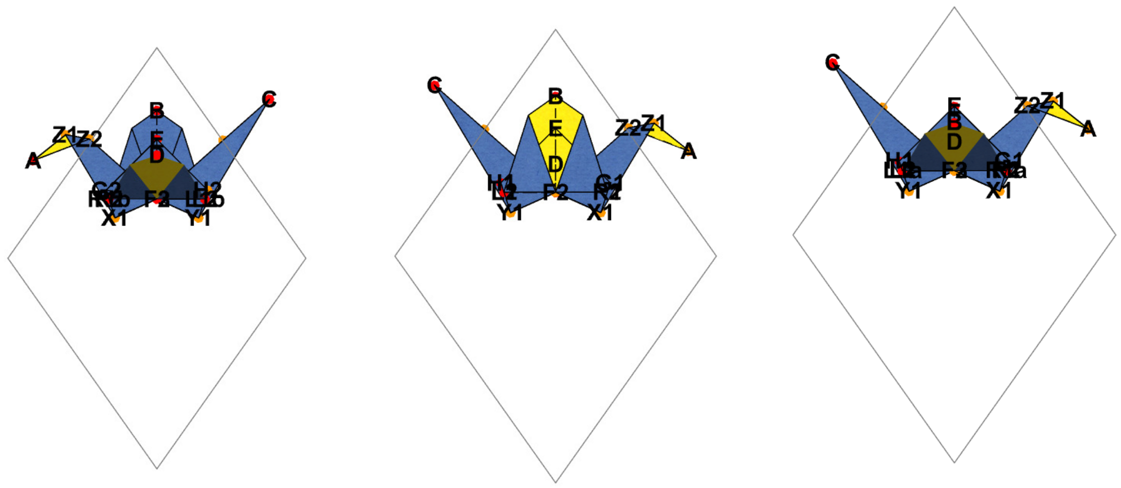
[1] T. Ida, H. Takahashi, A new modeling of classical folds in computational origami, in: P. Janičić, Z. Kovács (Eds.), Proceedings of the 13 th International Conference on Automated Deduction in Geometry, volume 352 of *EPTCS*, Elsevier Inc., 2021, pp. 41–53.

[2] H. Huzita (Ed.), Proceedings of the First International Meeting of Origami Science and Technology, Ferrara, Italy, 1989.

[3] J. Justin, Résolution par le pliage de l'équation du 3e degré et applications géométriques, L'Ouvert (1986) 9 – 19.

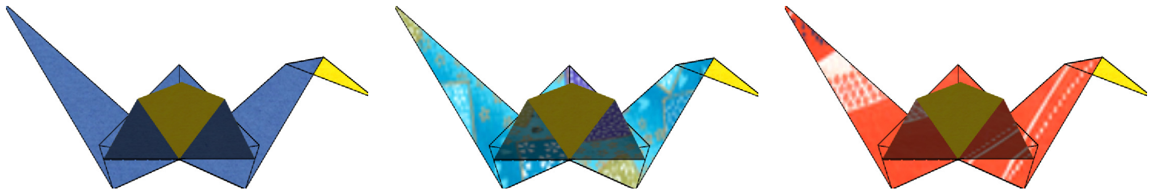
[4] K. Fushimi, M. Fushimi, Geometry of Origami, Nippon Hyoron sha Co. Ltd., 1979. (in Japanese).

[5] T. Ida, An introduction to Computational Origami, Texts and Monographs in Symbolic Computation, Springer International Publishing Switzerland, 2020.



(a) $\mathcal{O}_{69}$: by FO (b) $\mathcal{O}_{70}$: by turn over (c) $\mathcal{O}_{71}$: by FO

Figure 10: Opening wings



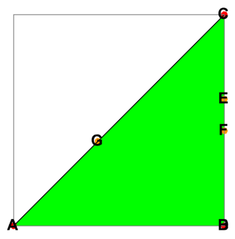
(a) final (b) textured crane (c) another textured crane

Figure 11: Monotone and textured cranes

[6] M. Bezem, J. W. Klop, Abstract reduction systems, volume Term Rewriting Systems, Cambridge University Press, 2003, pp. 7 – 23.
[7] T. Ida, D. Tepeneu, B. Buchberger, J. Robu, Proving and Constraint Solving in Computational Origami, in: Proceedings of the 7th International Symposium on Artificial Intelligence and Symbolic Computation (AISC 2004), volume 3249 of *Lecture Notes in Artificial Intelligence*, 2004, pp. 132–142.
[8] Wolfram Research, Inc., Mathematica, 2023.
[9] Eos project, 2024. URL: https://www.i-eos.org.

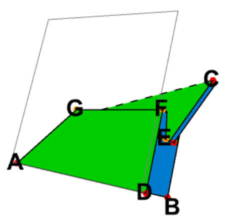
A. Classical folds realizable by cut-and-glue technique

- Squash fold
  SquashFold = InserReverseFold; (O1)
- Inside crimp pleat fold
- Outside crimp pleat fold

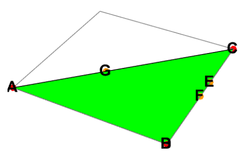


(a) before

$\looparrowright^*$

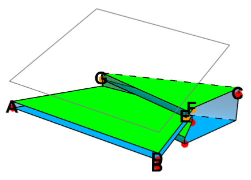


(b) after



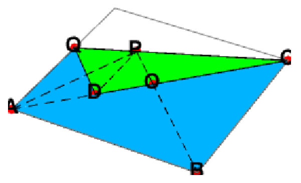
(a) before

$\looparrowright^*$



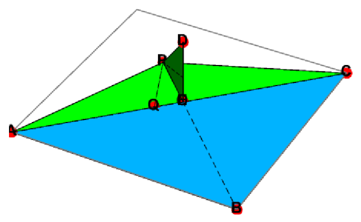
(b) after

- Rabbit ear fold



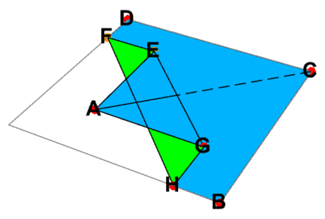
(a) before

$\looparrowright^*$



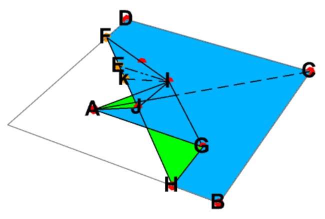
(b) after

- Swivel fold



(a) before

$\looparrowright^*$



(b) after

The geometry of N-body orbits and the DFT —Extended Abstract for work in progress —

Patrick D. F. Ion[1,2,*,†]

[1]International Mathematical Knowledge Trust, Mathematical Reviews ret'd
[2]University of Michigan, Ann Arbor, MI, USA

Keywords
Discrete Fourier Transform, N-body orbits,

1. Extended Abstract

The context is the Newtonian equal-mass three-body problem [Newton1687]. It's been a couple of decades since the discovery by Cris Moore [Moore1993] of a new periodic choreographic orbit, the first since Euler [Euler1767] and Lagrange [Lagrange1772]. Choreographic means that the all the particles follow the same orbital path. This figure-eight orbit was a numerical solution done on a Mac SE when looking for braids in orbits. The proof of its mechanical existence by Richard Montgomery [Montgomery1998] and Alain Chenciner [ChencinerMontgomery2000]was seen as important [see also [Chen2001] and [Nauenberg2007]] . Poincaré [Poincare1890] had discussed the necessarily complex, even chaotic, nature of 3-body orbits [Poincare1890] . This led to additional hundreds of new periodic choreographic orbits found numerically by Carles Simó [Simo2002] and later others [SuvakovDmitrasinovic2013]. The required proofs that these were also more than numerical objects still remain to be provided, with a few exceptions.

At about the same time, there was a renewal of interest in the use of the discrete Fourier transform (DFT) in Euclidean geometry. This subject goes back to Jesse Douglas [Douglas1940a] and Isaac Schoenberg [Schoenberg1950]. The second simplest consideration of this type is based on the harmonic analysis of the cyclic group of order 3 (second because order 2 is even simpler than 3). The basic assertion is then the classical construction of Napoleon's Theorem. Any triangle, seen as a triple of points in the complex plane, may be written as a complex linear combination of the totally degenerate triangle consisting of three coincident points located at 1, and the two standard equilateral triangles drawn in the unit disk with a vertex at 1, one for each possible orientation.

Returning to mechanics, one remarks that a solution of a three-body problem means giving the evolution in space of the three coordinates of the point masses involved. If the masses are all equal we're looking at the evolution of a simple triangle in the plane, thanks to the conservation laws of mechanics. Viewing the triangle in terms of harmonic coordinates as mentioned above, the first coordinate is the constant center of gravity of the three masses, so unmoving. Thus, to a 3-body solution correspond two more plane curves which are the tracks of the two non-degenerate harmonic coordinates.

In the case of the new figure-eight choreography, the DFT leads to two symmetrical 'triangular platelet boundaries'. It is known that the figure-eight orbit is not a lemniscate, or indeed parametrizable in terms of well-known special functions. So it might seem there may be some collection of special functions associated with Newtonian mechanics and good for parameterizing such curves.

It is appealing to see what the apparently very complicated higher-order Simó choreographies may lead to. One takes the conventional orbits and performs a DFT as above, then plots the resulting curves. These display visually a high degree of symmetry and regularity not apparent in the original orbits.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

pion@umich.edu (P. D. F. Ion)
https://websites.umich.edu/~pion/ (P. D. F. Ion)
0000-0002-4957-5812 (P. D. F. Ion)

Actually Simó's published discussions of how he found and calculated his 343 new periodic 3-body choreographies, and a number of choreographies for more bodies (some simple ones being just equally spaced rings of more than 3 particles) do not provide full sets of initial conditions that allow reproducing his results in, say, Octave (an open source analogue of Matlab). He remarks in his work that published initial values are often not precise enough to allow numerical following of orbits that are claimed, or indeed illustrated. So to produce the required DFT images I had to reverse engineer the (to me) rather odd plot format made public by Simó. The results I put up on a personal website at the University of Michigan [IonWeb]. Then I redid another version, creating SVG images using Mathematica 4, and added those for viewing.

Early on, there was much interest in recreating the original figure-eight orbit; many people did so. There were contributions from numerical analysis experts — such as Broucke [Broucke1975], Hadjidemetriou [Hadjidemetriou1975], Kapela et al. [Kapela2005]– and celestial orbit people — such as Marchal [Marchal2002], Hénon [Henon1976], Aarseth [Aarseth2003], Alexander D.Bruno, Montaldi and Steckles [MontaldiSteckles2013], Gerver [Gerver2003a], Moeckel [Moeckel2012], Terracini [Terracini2006], Ferrario [Ferrario2024], Zhifu Xie [Xie2022] — and also by others — such as Jenkins [JenkinsWeb], Vanderbei [VanderbeiWeb]; Jenkins, a self-proclaimed amateur, like others, also created a notable web site allowing orbit viewing using Java. The methods ranged from Runge-Kutta numerics of various types to action minimization and other variational routines, or used built-in solvers like those of [Mathematica], [Maple], or [Matlab] and [Octave]. At one point I counted about 40 different approaches. Of course, a number of the web presences of these efforts have by now disappeared. Notable to me was that though there were lots of figure eights, say, there was no clarity that they were all describing the same orbit—the results are given as a finite sequence of computed coordinate values of widely varying precisions. Phil Sharp [Sharp2006] (and I) produced a Matlab routine that showed the choreographic eight, but a change of 1 part in 10^{12} in initial conditions splits the result into three parallel orbits that were, of course, visually indistinguishable, if plotted ordinarily, from the true choreography's single repeated orbit.

More recently, in 2019, Li and Liao [LiLao2019] announced discovery of 313 more periodic collisionless orbits. Then in 2023 Hristov, Hristova, Dmitrašinović and Tanikawa [HristovEtAl2024] announced more than 12,000 distinct 3-body orbits, derived using newer computing hardware and a refined assignment of symbol sequences to trajectories that made search for suitable orbits easier. They also pointed out some edge problems with Li and Liao's listing. It is now time to examine the new orbits from the DFT point of view. This involves reviving some older constructions which ran fine under earlier versions of scripting languages (e.g. Python, Javascript), graphics technology (e.g. SVG), numerical technology (e.g., [Octave], [Numpy] etc., Java) and symbolic computation platforms (e.g. [Mathematica] and [Maple]).

Acknowledgments

Mathematical Reviews and the University of Michigan have provided unparalleled access to the literature of mathematics and software.

References

[Aarseth2003] Aarseth, Sverre J. *Gravitational N-body simulations. Tools and algorithms.* Cambridge Monographs on Mathematical Physics. Cambridge: Cambridge University Press (ISBN 0-521-43272-3/hbk). xv, 413 p. (2003). Zbl 1098.70001

[Broucke1975] Broucke, Roger; *On relative periodic solutions of the planar general three-body problem*, Celestial Mechanics 12, 439-462 (1975).

[BrouckeBoggs1975] Broucke, Roger; Boggs, D., *Periodic orbits in the planar general three-body problem*, Celest Mech, 11, 13-38 (1975) · Zbl 0303.70015

[Chen2001] Chen, Kuo-Chang, *On Chenciner-Montgomery's orbit in the three-body problem.*, Discrete Contin. Dyn. Syst., 7, No. 1, 85–90, (2001), DOI: 10.3934/dcds.2001.7.85, Zbl:1093.70502

[ChencinerGerverEtAl2002] Chenciner, Alain; Gerver, Joseph; Montgomery, Richard; Simó, Carles *Simple choreographic motions of N bodies: a preliminary study.* Newton, Paul (ed.) et al., Geometry, mechanics, and dynamics. Volume in honor of the 60th birthday of J. E. Marsden. New York, NY: Springer (ISBN 0-387-95518-6/hbk). 287-308 (2002).Zbl 1146.70333

[ChencinerMontgomery2000] Chenciner, A.; Montgomery, R., *A remarkable periodic solution of the three-body problem in the case of equal masses*, Ann Math, 152, 3, 881-901 (2000) · Zbl 0987.70009

[Douglas1940a] Douglas, Jesse, *Geometry of polygons in the complex plane.* J. Math. Phys. Mass. Inst. Tech. **19** (1940), 93–130. MR:0001574

[Douglas1940b] Douglas, Jesse, *On linear polygon transformations.* Bull. Amer. Math. Soc. **46** (1940), 551–560. MR:0002178

[Douglas1960] Douglas, Jesse, *A theorem on skew pentagons.* Scripta Math. 25 (1960) 5–9. MR:0117643

[Euler1767] Euler, Leonhard, *De motu rectilineo trium corporum se mutuo attrahentium*, Novi commentarii academiae scientiarum Petropolitanae 11, 144-151 (1767).

[Ferrario2024] Ferrario, Davide L. *Symmetries and periodic orbits for the n-body problem: about the computational approach.* Preprint, arXiv:2405.07737 [math.CA] (2024).

[Ferrario2020] Ferrario, Davide L. *Fixed points and the inverse problem for central configurations.* Topol. Methods Nonlinear Anal. 56, No. 2, 579-588 (2020). Zbl 1476.55007

[Gerver2003a] Gerver, Joseph L., *Noncollision singularities in the n-body problem.* Dynamical systems. Part I. Hamiltonian systems and celestial mechanics. Selected papers from the Research Trimester held in Pisa, Italy, February 4–April 26, 2002. Pisa: Scuola Normale Superiore. Pubblicazioni del Centro di Ricerca Matematica Ennio de Giorgi. Proceedings, 57-86 (2003). Zbl 1318.70008

[Gerver2003b] Gerver, Joseph L. *Noncollision singularities: do four bodies suffice?* Exp. Math. 12, No. 2, 187-198 (2003). Zbl 1254.70027

[Hadjidemetriou1975] J. D. Hadjidemetriou, J. D., *The stability of periodic orbits in the three-body problem*, Celestial Mechanics 12, 255-276 (1975).

[HadjidemetriouChristides1975] Hadjidemetriou, J. D. , and Christides, T. , *Families of periodic orbits in the planar three-body problem*, Celestial mechanics 12, 175-187 (1975).

[Henon1976] Hénon, Marcel, *A family of periodic solutions of the planar three-body problem, and their stability*, Celestial mechanics 13, 267-285 (1976).

[Henon1977] Hénon, Marcel, *Stability of interplay motions*, Celestial mechanics 15, 243-261 (1977).

[HristovEtAl2024] Hristov, Ivan; Hristova, Radoslava; Dmitrašinović, Veljko; Tanikawa, Kiyotaka *Three-body periodic collisionless equal-mass free-fall orbits revisited.* (English) Zbl 07834263 Celest. Mech. Dyn. Astron. 136, No. 1, Paper No. 7, 20 p. (2024). MSC: 70F07 70-08 arXiv preprint ; associated data files

[Kapela2007] Kapela, Tomasz and Simó, Carles, *Computer assisted proofs for nonsymmetric planar choreographies and for stability of the Eight*, Nonlinearity, 20, No. 5, 1241–1255, (2007), doi: 0.1088/0951-7715/20/5/010, Zbl:1115.70008

[Kapela2005] Kapela, Tomasz, *N-body choreographies with a reflectional symmetry – computer-assisted existence proofs*, EQUADIFF 2003. Proceedings of the international conference on differential equations, Hasselt, Belgium, July 22–26, 2003, 999–1004, (2005), Hackensack, NJ: World Scientific, Zbl:1116.70019

[Kapela2003] Kapela, Tomasz and Zgliczyński, Piotr, *The existence of simple choreographies for the N-body problem – a computer-assisted proof*, Nonlinearity, 16, No. 6, 1899–1918,(2003), doi:10.1088/0951-7715/16/6/302, Zbl:1060.70023

[Lagrange1772] Lagrange, Jean Louis, *"Essai sur le probleme des trois corps*, Prix de l'Academie Royale des Sciences de Paris 9, 292 (1772).

[LiLao2017] X. Li and S. Liao, *More than six hundred new families of Newtonian periodic planar collisionless three-body orbits*, SCIENCE CHINA Physics, Mechanics & Astronomy 60, 129511 (2017). arXiv:1705.00527v4

[LiJingLao2018] X. Li, Y. Jing, and S. Liao, *Over a thousand new periodic orbits of a planar three-body system with unequal masses*, Publications of the Astronomical Society of Japan 00, 1-7 (2018). arXiv:1709.04775

[LiLiLao2021] X. Li, X. Li and S. Liao, *One family of 13315 stable periodic orbits of non-hierarchical unequal-mass triple systems*, SCIENCE CHINA Physics, Mechanics & Astronomy 64, 219511 (2021). arXiv:2007.10184

[LiaoLiYang2022] S. Liao, X. Li and Y. Yang, *Three-body problem - from Newton to supercomputer plus machine learning*, New Astronomy 96, 101850 (2022). arXiv:2106.11010v2

[LiLao2019] Li, Xiaomong, Liao, Shojun *Collisionless periodic orbits in the free-fall three-body problem.* New Astron. 70, 22-26 (2019). arXiv:1805.07980v1; https://doi.org/10.1016/j.newast.2019.01.003

[Marchal2002] Marchal, C., *How the method of minimization of action avoids singularities. Modern celestial mechanics: from theory to applications* (Rome, 2001), Celest Mech Dyn Astron, 83, 1-4, 325-353 (2002) · Zbl 1073.70011

[Moeckel2012] Moeckel, R.; Montgomery, R.; Venturelli, A., *From brake to syzygy*, Arch. Ration. Mech. Anal., 204, 1009-1060 (2012) · Zbl 1286.70014 · doi:10.1007/s00205-012-0502-y

[MontaldiSteckles2013] Montaldi, James and Steckles, Katrina, *Classification of symmetry groups for planar n-body choreographies*, Forum Math. Sigma, **1**, 55, Id/No e5, (2013), doi:10.1017/fms.2013.5, Zbl 1325.37017

[Montgomery1998] Montgomery, R., *The N-body problem, the braid group, and action-minimizing periodic solutions*, Nonlinearity, 11, 2, 363-376 (1998) · Zbl 1076.70503 · doi:10.1088/0951-7715/11/2/011

[Montgomery2007] Montgomery, R., *The zero angular momentum, three-body problem: all but one solution has syzygies*, Ergod. Theory Dyn. Syst., 27, 6, 1933-1946 (2007) · Zbl 1128.70005 · doi:10.1017/S0143385707000338

[Montgomery2023] Montgomery, R.: *Dropping bodies.* Math.Intell. 1-7. (2023)

[Moore1993] Moore, Cristopher *Braids in classical dynamics.* Phys. Rev. Lett. 70, No. 24, 3675-3679 (1993). Zbl:1050.37522

[Nauenberg2007] Nauenberg, Michael *Continuity and stability of families of figure eight orbits with finite angular momentum.* (English) Zbl 1162.70009 Celest. Mech. Dyn. Astron. 97, No. 1, 1-15 (2007).

[Newton1687] Newton, Isaac *Philosophiae naturalis principia mathematica* (London: Royal Society Press, 1687).

[Poincare1890] Poincaré, Jeam Henri, *Sur le probleme des trois corps et les equations de la dynamique*, Acta Mathematica 13, 1-271 (1890).

[Schoenberg1950] Schoenberg, Isaac Jacob, *The finite Fourier series and elementary geometry.* Amer. Math. Monthly **57** (1950), 390–404. MR:0036332 (12,92f)

[Schoenberg1981] Schoenberg, Isaac Jacob, *The harmonic analysis of skew polygons as a source of outdoor sculptures.* The geometric vein, pp. 165–176, Springer, New York-Berlin, 1981. MR:0661776

[Schoenberg1982] Schoenberg, Isaac Jacob, *Mathematical time exposures.* Mathematical Association of America, Washington, DC, 1982. ix+270 pp. ISBN: 0-88385-438-4. MR:0711022

[Sharp2004] Sharp, P. W., *Comparisons of integrators on a diverse collection of restricted three-body test problems.* IMA J. Numer. Anal. 24, No. 4, 557-575 (2004). Zbl 1059.70002

[Sharp2006] Sharp, P. W., *N-body simulations: the performance of some integrators.* ACM Trans. Math. Softw. 32, No. 3, 375-395 (2006). Zbl 1230.70004

[Sharp2019] Sharp, P. W., *The performance of the N-body integrator SSS.* (Numer. Algorithms 81, No. 4, 1459-1472 (2019). Zbl 1416.70006

[Simo2002] Simó, Carles, *Dynamical properties of the figure eight solution of the three-body problem*, Contemp Math, 292, 209-228 (2002) · Zbl 1151.70316

[SuvakovDmitrasinovic2013] Šuvakov , M., and Dmitrašinović, V., *Three classes of Newtonian three-body planar periodic orbits*, Physical Review Letters 110, 114301 (2013).

[Terracini2006] Terracini, Susanna, *On the variational approach to the periodic n-body problem.* Celest. Mech. Dyn. Astron. 95, No. 1-4, 3-25 (2006). Zbl 1219.70030

[Xie2022] Xie, Zhifu *Remarks on the inverse problem of the collinear central configurations in the N-body problem.* Electron. Res. Arch. 30, No. 7, 2540-2549 (2022). Zbl 1522.70019

A. Online Resources

References

[IonFeat] Ion, Patrick D. F., *Geometry and the Discrete Fourier Transform* (2010) on AMS site with link deficiencies ; modernized working copy on Michigan site

[IonWeb] Ion, Patrick D. F., *Home page leading to N-body results* (2003-2024) website

[JenkinsWeb] Jenkins, Bob, *Home Page including sections on Space with non-colliding orbits, Javascript Canvas for Gravitational Orbit Simulation* website

[Maple] Maple (Version 15) website

[Matlab] Matlab website

[Mathematica] Wolfram Mathematica (Version 11.1 used); now 14.1 website

[Numpy] Numpy website

[Octave] Octave website

[VanderbeiWeb] Vanderbei, Robert J., *n-body orbits gallery of varying types* (ca. 2006–present); Stable Solutions to the Planar Three-Body Problem; The Šuvakov-Dmitrašinović Suite website

Gröbner basis computation via learning

Hiroshi Kera[1,*], Yuki Ishihara[2], Tristan Vaccon[3] and Kazuhiro Yokoyama[4]

[1]*Chiba University, 1-33 Yayoi-cho, Inage-ku, Chiba-shi, Chiba, 2638522, Japan*
[2]*Nihon University, 1-8-14 Kanda Surugadai, Chiyoda-ku, Tokyo, 1018308, Japan*
[3]*Université de Limoges; CNRS, XLIM, UMR 7252, Limoges, France*
[4]*Rikkyo University, 3-34-1, Nishi-Ikebukuro, Toshima-ku, Tokyo, 1718501, Japan*

Abstract
Solving a polynomial system, or computing an associated Gröbner basis, has been a fundamental task in computational algebra. However, it is also known for its notorious doubly exponential time complexity in the number of variables in the worst case. This paper is the first to address the learning of Gröbner basis computation with Transformers. The training requires many pairs of a polynomial system and the associated Gröbner basis, raising two novel algebraic problems: random generation of Gröbner bases and transforming them into non-Gröbner ones, termed as backward Gröbner problem. We resolve these problems with 0-dimensional radical ideals, the ideals appearing in various applications. The experiments show that our dataset generation method is at least three orders of magnitude faster than a naive approach, overcoming a crucial challenge in learning to compute Gröbner bases, and Gröbner computation is learnable in a particular class.

Keywords
Gröbner Bases, Machine Learning, Transformer

1. Introduction

Understanding the properties of polynomial systems and solving them have been a fundamental problem in computational algebra and algebraic geometry with vast applications in cryptography [1, 2], control theory [3], statistics [4, 5], computer vision [6], systems biology [7], and so forth. Special sets of polynomials called Gröbner bases [8] play a key role to this end. In linear algebra, the Gaussian elimination simplifies or solves a system of linear equations by transforming its coefficient matrix into the reduced row echelon form. Similarly, a Gröbner basis can be regarded as a reduced form of a given polynomial system, and its computation is a generalization of the Gaussian elimination to general polynomial systems. However, computing a Gröbner basis is known for its notoriously bad computational cost in theory and practice. It is an NP-hard problem with the doubly exponential worst-case time complexity in the number of variables [9, 10]. Nevertheless, because of its importance, various algorithms have been proposed in computational algebra to obtain Gröbner bases in better runtime. Examples include Faugère's F4/F5 algorithms [11, 12] and M4GB [13].

In this study, we investigate Gröbner basis computation from a learning perspective, envisioning it as a practical compromise to address large-scale polynomial system solving and understanding, where mathematical algorithms are computationally intractable. The learning approach does not require explicit design of computational procedures, and we only need to train a model using a large amount of (non-Gröbner set, Gröbner basis) pairs. Further, if we restrict ourselves to a particular class of Gröbner bases (or associated *ideals*), the model may internally find some patterns useful for prediction. The success of learning indicates the existence of such patterns, which encourages the improvement of mathematical algorithms and heuristics. Several recent studies have already addressed

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
*Corresponding author.

kera@chiba-u.jp (H. Kera); ishihara.yuki@nihon-u.ac.jp (Y. Ishihara); tristan.vaccon@unilim.fr (T. Vaccon); kazuhiro@rikkyo.ac.jp (K. Yokoyama)
https://hkera.wordpress.com (H. Kera); https://researchmap.jp/yishihara (Y. Ishihara); https://www.unilim.fr/pages_perso/tristan.vaccon/ (T. Vaccon)
0000-0002-9830-0436 (H. Kera); 0000-0003-4057-3703 (Y. Ishihara); 0000-0003-4208-8349 (T. Vaccon); 0000-0002-5072-7799 (K. Yokoyama)

mathematical tasks via learning, particularly using Transformers [14, 15, 16]. For example, [14] showed that Transformers can learn symbolic integration simply by observing many $(\mathrm{d}f/\mathrm{d}x, f)$ pairs in training. The training samples are generated by first randomly generating f and computing its derivative $\mathrm{d}f/\mathrm{d}x$ and/or by the reverse process.

However, a crucial challenge in the learning of Gröbner basis computation is that it is mathematically unknown how to efficiently generate many (non-Gröbner set, Gröbner basis) pairs. We need an efficient backward approach (i.e., *solution-to-problem* computation) because, as discussed above, the forward approach (i.e., *problem-to-solution* computation) is prohibitively expensive. To this end, we frame two problems: (i) a random generation of Gröbner bases and (ii) a backward transformation from a Gröbner basis to an associated non-Gröbner set. To our knowledge, neither of them has been addressed in the study of Gröbner bases because of the lack of motivations; all the efforts have been dedicated to the forward computation from a non-Gröbner set to Gröbner basis.

Tackling aforementioned two unexplored algebraic problems, we investigates the first learning approach to the Gröbner computation using Transformers and experimentally show its learnability uncovered two unexplored algebraic problems in the 0-dimensional case. Our experiments show that the proposed dataset generation is highly efficient and faster than a baseline method by three or four orders of magnitude. Further, we observe a learnability gap between polynomials on finite fields and infinite fields while predicting polynomial supports are more tractable. Full version of this paper can be found in [17].

2. New algebraic problems for dataset generation

Our notations and definitions follow [18] except that we call power products of indeterminate terms instead of monomials. By Gröbner basis computation, we mean computation of reduced Gröbner bases. Our goal is to realize Gröbner basis computation through learning. To this end, we need a large training set $\{(F_i, G_i)\}_{i=1}^m$ with finite polynomial set $F_i \subset k[x_1, \dots, x_n]$ and Gröbner basis G_i of ideal $\langle F_i \rangle$. As the computation from F_i to G_i is computationally expensive in general, we instead resort to *backward generation* (i.e., solution-to-problem process); that is, we generate a Gröbner basis G_i randomly and transform it to non-Gröbner set F_i.

Problem 2.1 (Random generation of Gröbner bases). *Find a collection $\mathscr{G} = \{G_i\}_{i=1}^m$ with the reduced Gröbner basis $G_i \subset k[x_1, \dots, x_n]$ of $\langle G_i \rangle$, $i = 1, \dots, m$. The collection should contain diverse bases, and we need an efficient algorithm for constructing them.*

Problem 2.2 (Backward Gröbner problem). *Given a Gröbner basis $G \subset k[x_1, \dots, x_n]$, find a collection $\mathscr{F} = \{F_i\}_{i=1}^{\mu}$ of polynomial sets that are not Gröbner bases but $\langle F_i \rangle = \langle G \rangle$ for $i = 1, \dots, \mu$. The collection should contain diverse sets, and we need an efficient algorithm for constructing them.*

Problems. 2.1 and 2.2 require the collections $\mathscr{G}, \mathscr{F}$ to contain diverse polynomial sets. Thus, the algorithms for these problems should not be deterministic but should have some controllable randomness.

What makes the learning of Gröbner basis computation hard is that, to our knowledge, neither (i) a random generation of Gröbner basis nor (ii) the backward transform from Gröbner basis to non-Gröbner set has been considered in computational algebra. Its primary interest has been instead posed on Gröbner basis computation (i.e., forward generation), and nothing motivates the random generation of Gröbner basis nor the backward transform. Interestingly, machine learning now sheds light on them. Formally, we address the following problems for dataset generation.

In this paper, we tackle these problems in the case of radical 0-dimensional ideals. We first address Prob. 2.1 using the fact that 0-dimensional radical ideals are generally *in shape position.*

Definition 2.3 (Shape position). Ideal $I \subset k[x_1, \dots, x_n]$ is called in *shape position* if some univariate polynomials $h, g_1, \dots, g_{n-1} \in k[x_n]$ form the reduced $\prec_{\mathrm{lex}}$-Gröbner basis of I as follows.

$$G = \{h, x_1 - g_1, \dots, x_{n-1} - g_{n-1}\}. \tag{2.1}$$

Particularly, 0-dimensional radical ideals are almost always in shape position if k is an infinite field or finite field with large field order [19, 20]. With this fact, an efficient sampling of Gröbner bases of 0-dimensional radical ideals can be realized by sampling n polynomials in $k[x_n]$, i.e., $h, g_1, \dots, g_{n-1}$ with $h \neq 0$. We have to make sure that the degree of h is always greater than that of $g_1, \dots, g_{n-1}$, which is necessary and sufficient for G to be a reduced Gröbner basis. This approach involves efficiency and randomness, and thus resolving Prob. 2.1. To address Prob. 2.2, we consider the following problem.

Problem 2.4. *Let $I \subset k[x_1, \dots, x_n]$ be a 0-dimensional ideal, and let $G = (g_1, \dots, g_t)^\top \in k[x_1, \dots, x_n]^t$ be its $\prec$-Gröbner basis with respect to term order $\prec$.[1] Find a polynomial matrix $A \in k[x_1, \dots, x_n]^{s\times t}$ giving a non-Gröbner set $F = (f_1, \dots, f_s)^\top = AG$ such that $\langle F\rangle = \langle G\rangle$.*

Namely, we generate a set of polynomials $F = (f_1, \dots, f_s)^\top$ from $G = (g_1, \dots, g_t)^\top$ by $f_i = \sum_{j=1}^{t} a_{ij} g_j$ for $i = 1, \dots, s$, where $a_{ij} \in k[x_1, \dots, x_n]$ denotes the (i, j)-th entry of A. Note that $\langle F\rangle$ and $\langle G\rangle$ are generally not identical, and the design of A such that $\langle F\rangle = \langle G\rangle$ is of our question.

A similar question was studied without the Gröbner condition in [21, 22]. They provided an algebraic necessary and sufficient condition for the polynomial system of F to have a solution outside the variety defined by G. This condition is expressed explicitly by multivariate resultants. However, strong additional assumptions are required: A, F, G are homogeneous, G is a regular sequence, and in the end, $\langle F\rangle = \langle G\rangle$ is only satisfied up to saturation. Thus, they are not compatible with our setting and method for Prob. 2.1. Our analysis gives the following results for the design A to achieve $\langle F\rangle = \langle G\rangle$ for the 0-dimensional case.

Theorem 2.5. *Let $G = (g_1, \dots, g_t)^\top$ be a Gröbner basis of a 0-dimensional ideal in $k[x_1, \dots, x_n]$. Let $F = (f_1, \dots, f_s)^\top = AG$ with $A \in k[x_1, \dots, x_n]^{s\times t}$.*

1. *If $\langle F\rangle = \langle G\rangle$, it implies $s \geq n$.*
2. *If A has a left-inverse in $k[x_1, \dots, x_n]^{t\times s}$, $\langle F\rangle = \langle G\rangle$ holds.*
3. *The equality $\langle F\rangle = \langle G\rangle$ holds if and only if there exists a matrix $B \in k[x_1, \dots, x_n]^{t\times s}$ such that each row of $BA - E_t$ is a syzygy of G, where E_t is the identity matrix of size t.*

We now assume $\prec = \prec_{\text{lex}}$ and 0-dimensional ideals in shape position. Then, G has exactly n generators. When $s = n$, we have the following.

Proposition 2.6. *For any $A \in k[x_1, \dots, x_n]^{n\times n}$ with $\det(A) \in k \setminus \{0\}$, we have $\langle F\rangle = \langle G\rangle$.*

As non-zero constant scaling does not change the ideal, we focus on A with $\det(A) = \pm 1$ without loss of generality. Such A can be constructed using the Bruhat decomposition $A = U_1 P U_2$, where $U_1, U_2 \in \mathrm{ST}(n, k[x_1, \dots, x_n])$ are upper-triangular matrices with all-one diagonal entries (i.e., unimodular upper-triangular matrices) and $P \in \{0, 1\}^{n\times n}$ denotes a permutation matrix. Noting that A^{-1} satisfies $A^{-1}A = E_n$, we have $\langle AG\rangle = \langle G\rangle$ from Thm. 2.5. Therefore, random sampling (U_1, U_2, P) of unimodular upper-triangular matrices U_1, U_2 and a permutation matrix P resolves the backward Gröbner problem for $s = n$. We extend this idea to the case of $s > n$ using a rectangular unimodular upper-triangular matrix $U_2 = \begin{pmatrix} U_2' \\ O_{s-n,n} \end{pmatrix} \in k[x_1, \dots, x_n]^{s\times n}$, where $U_2' \in k[x_1, \dots, x_n]^{n\times n}$ is a unimodular upper-triangular matrix and $O_{s-n,n} \in k[x_1, \dots, x_n]^{(s-n)\times n}$ is the zero matrix. The permutation matrix is now $P \in \{0, 1\}^{s\times s}$. Our strategy is to compute $F = U_1 P U_2 G$, which only requires a sampling of $\mathcal{O}(s^2)$ polynomials in $k[x_1, \dots, x_n]$, and $\mathcal{O}(n^2 + s^2)$-times multiplications of polynomials.

3. Experiments

We present the efficiency of our dataset generation method and the learnability of Gröbner basis computation. The experiments were conducted with 48-core CPUs, 768GB RAM, and NVIDIA RTX A6000ada GPUs. Due to the space limitation, we cannot present full experimental setup. See the full version in [17].

[1]We surcharge notations to mean that the set $\{g_1, \dots, g_t\}$ defined by the vector G is a $\prec$-Gröbner basis.

Table 1
Runtime comparison (in seconds) of forward generation (F.) and backward generation (B.) of dataset $\mathscr{D}_n(\mathbb{F}_7)$ of size 1,000. The forward generation used either of the three algorithms provided in SageMath with the libSingular backend. We set a timeout limit to five seconds (added to the total runtime at every occurrence) for each Gröbner basis computation. The numbers with † and ‡ include the timeout for more than 13 % and 24 % of the runs, respectively.

Method	$n = 2$	$n = 3$	$n = 4$	$n = 5$
F. (STD)	4.65	129	873†	1354‡
F. (SLIMGB)	4.67	149	712†	1259‡
F. (STDFGLM)	5.78	12.6	44.2	360
B. (ours)	**.003**	**.005**	**.009**	**.014**

Dataset generation. We constructed 12 datasets $\mathscr{D}_n(k)$ for $n \in \{2, 3, 4, 5\}$ and $k \in \{\mathbb{F}_7, \mathbb{F}_{31}, \mathbb{Q}\}$ and measured the runtime of our backward generation and naive forward generation (i.e., Gröbner basis computation). In the backward generation, we sampled Gröbner bases of ideals in shape position. In this step, univariable polynomials were generically sampled in $k[x_1, \ldots, x_n]_{\leq 5}$. Next, Gröbner bases were transformed to non-Gröbner sets based on Thm. 2.5. Random polynomials in Bruhat decomposition (i.e., U_1 and U_2') were sampled from $k[x_1, \ldots, x_n]_{\leq 3}$ and restricted to monomials and binomials. For $\mathbb{Q}$, coefficients of all sampled polynomials were bounded as a/b with $a, b \in \{-5, \ldots, 5\}$ and we only accepted F with coefficients such as $a, b \in \{-100, \ldots, 100\}$. This restriction is required from our machine learning model and learning framework. For forward generation, we adopted three algorithms given by SageMath [23] with the libSingular backend. For a fair comparison, forward generation computed Gröbner bases of the non-Gröbner sets given by the backward generation, leading to the identical dataset. As Tab. 1 shows, our backward generation is significant orders of magnitude faster than the forward generation. A sharp runtime growth is observed in the forward generation as the number of variables increases. Note that these numbers only show the runtime on 1,000 samples, while training typically requires millions of samples. Therefore, the forward generation is almost infeasible, and the proposed method resolves a bottleneck in the learning of Gröbner basis computation.

Learning results. We used a standard Transformer (e.g., 6 encoder/decoder layers and 8 attention heads) and a standard training setup. The batch size was set to 16, and models were trained for 8 epochs. Each polynomial set in the datasets is converted into a sequence using the prefix representation and the separator tokens. To make the input sequence length manageable for vanilla Transformers, we used simpler datasets $\mathscr{D}_n^-(k)$ using U_1, U_2' of a moderate density $\sigma \in (0, 1]$. This makes the maximum sequence length less than 5,000. Specifically, we used $\sigma = 1.0, 0.6, 0.3, 0.2$ for $n = 2, 3, 4, 5$, respectively. The training set has one million samples, and the test set has one thousand samples. Table 2 shows that trained Transformers successfully compute Gröbner bases with moderate/high accuracy. Not shown here, but we found several examples in the datasets for which Transformer successfully compute Gröbner bases significantly faster than math algorithms. The accuracy shows that the learning is more successful on infinite field coefficients $k \in \{\mathbb{Q}, \mathbb{R}\}$ than finite field ones $k = \mathbb{F}_p$. This may be a counter-intuitive observation because there are more possible coefficients in G and F for $\mathbb{Q}$ than $\mathbb{F}_p$. Specifically, for G, the coefficient $a/b \in \mathbb{Q}$ is restricted to those with $a, b \in \{-5, \ldots, 5\}$ (i.e., roughly 50 choices), and $a, b \in \{-100, \ldots, 100\}$ (i.e., roughly 20,000 choices) for F. In contrast, there are only p choices for $\mathbb{F}_p$. The performance even degrades for the larger order $p = 31$. Interestingly, the support accuracy shows that the terms forming the polynomial (i.e., the *support* of polynomial) are correctly identified well. Thus, Transformers have difficulty determining the coefficients in finite fields. Several studies have also reported that learning to solve a problem involving modular arithmetic may encounter some difficulties [24, 25, 26].

Table 2
Accuracy [%] / support accuracy [%] of Gröbner basis computation by Transformer on $\mathscr{D}_n^-(k)$. In the support accuracy, two polynomials are considered identical if they consist of an identical set of terms (i.e., identical *support*), Note that the datasets for $n = 3, 4, 5$ are here constructed using U_1, U_2' with density $\sigma = 0.6, 0.3, 0.2$, respectively.

Ring	$n = 2, \sigma = 1$	$n = 3, \sigma = 0.6$	$n = 4, \sigma = 0.3$	$n = 5, \sigma = 0.2$
$\mathbb{Q}[x_1, \dots, x_n]$	94.6 / 97.9	96.1 / 98.6	96.2 / 98.6	91.8 / 97.9
$\mathbb{F}_7[x_1, \dots, x_n]$	66.6 / 76.6	78.8 / 87.6	80.9 / 91.1	83.2 / 91.4
$\mathbb{F}_{31}[x_1, \dots, x_n]$	44.7 / 82.7	58.5 / 89.3	73.9 / 93.9	80.0 / 93.4

4. Conclusion

This study proposed the first learning approach to a fundamental algebraic task, the Gröbner basis computation. While various recent studies have reported the learnability of mathematical problems by Transformers, we addressed the first problem with nontriviality in the dataset generation. Ultimately, the learning approach may be useful to address large-scale problems that cannot be approached by Gröbner basis computation algorithms because of their computational complexity. Transformers can output predictions in moderate runtime. The outputs may be incorrect, but there is a chance of obtaining a hint of a solution, as shown in our experiments. We believe that our study reveals many interesting open questions to achieve Gröbner basis computation learning.

Acknowledgments

This research was supported by JST ACT-X Grant Number JPMJAX23C8 and JSPS KAKENHI Grant Number JP22K13901. Yuta Kambe (Mitsubishi Electric Information Technology R&D Center) is not included in the authors due to a technical reason at submission.

References

[1] G. V. Bard, Algorithms for Solving Polynomial Systems, Springer US, 2009.
[2] T. Yasuda, X. Dahan, Y.-J. Huang, T. Takagi, K. Sakurai, MQ challenge: hardness evaluation of solving multivariate quadratic problems, Cryptology ePrint Archive (2015).
[3] H. Park, G. Regensburger (Eds.), Gröbner Bases in Control Theory and Signal Processing, De Gruyter, 2007.
[4] P. Diaconis, B. Sturmfels, Algebraic algorithms for sampling from conditional distributions, The Annals of Statistics 26 (1998) 363 – 397.
[5] T. Hibi, Gröbner bases. Statistics and software systems., Springer Tokyo, 2014.
[6] H. Stewenius, Gröbner Basis Methods for Minimal Problems in Computer Vision, Ph.D. thesis, Mathematics (Faculty of Engineering), 2005.
[7] R. Laubenbacher, B. Sturmfels, Computer algebra in systems biology, American Mathematical Monthly 116 (2009) 882–891.
[8] B. Buchberger, Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal), Ph.D. thesis, Mathematical Institute, University of Innsbruck, Austria, 1965. English translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475–511, 2006.
[9] E. W. Mayr, A. R. Meyer, The complexity of the word problems for commutative semigroups and polynomial ideals, Advances in Mathematics 46 (1982) 305–329.
[10] T. W. Dubé, The structure of polynomial ideals and Gröbner bases, SIAM Journal on Computing 19 (1990) 750–773.

[11] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases (F4), Journal of Pure and Applied Algebra 139 (1999) 61–88.

[12] J.-C. Faugère, A new efficient algorithm for computing Gröbner bases without reduction to zero (F5), in: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation, ISSAC '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 75–83.

[13] R. H. Makarim, M. Stevens, M4GB: An efficient Gröbner-basis algorithm, in: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC'17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 293–300.

[14] G. Lample, F. Charton, Deep learning for symbolic mathematics, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=S1eZYeHFDS.

[15] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, G. Parascandolo, Neural symbolic regression that scales, in: Proceedings of the 38th International Conference on Machine Learning, volume 139, 2021, pp. 936–945.

[16] F. Charton, Linear algebra with transformers, Transactions on Machine Learning Research (2022).

[17] H. Kera, Y. Ishihara, Y. Kambe, T. Vaccon, K. Yokoyama, Learning to compute gröbner bases, 2024. arXiv:2311.12904.

[18] D. A. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, Undergraduate Texts in Mathematics, Springer International Publishing, 2015.

[19] P. Gianni, T. Mora, Algebraic solution of systems of polynomial equations using Groebner bases, in: Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Springer Berlin Heidelberg, Berlin, Heidelberg, 1989, pp. 247–257.

[20] M. Noro, K. Yokoyama, A modular method to compute the rational univariate representation of zero-dimensional ideals, Journal of Symbolic Computation 28 (1999) 243–263.

[21] L. Busé, M. Elkadi, B. Mourrain, Resultant over the residual of a complete intersection, Journal of Pure and Applied Algebra 164 (2001) 35–57.

[22] L. Busé, Étude du résultant sur une variété algébrique, Theses, Université Nice Sophia Antipolis, 2001.

[23] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 10.0), 2023. https://www.sagemath.org.

[24] A. Power, Y. Burda, H. Edwards, I. Babuschkin, V. Misra, Grokking: Generalization beyond overfitting on small algorithmic datasets, arXiv abs/2201.02177 (2022).

[25] F. Charton, Can transformers learn the greatest common divisor?, arXiv abs/2308.15594 (2023).

[26] A. Gromov, Grokking modular arithmetic, arXiv abs/2301.02679 (2023).

Solving Estimation Problems Using Minimax Polynomials and Gröbner Bases[⋆]

Kenta Kuramochi[1], Akira Terui[2,*] and Masahiko Mikawa[3]

[1]*Master's Program in Mathematics, Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8571, Japan*

[2]*Institute of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan*

[3]*Institute of Library, Information and Media Science, University of Tsukuba, Tsukuba 305-8550, Japan*

Abstract

We propose a method for solving the speech direction estimation problem by computer algebra. The method is based on the function approximation using the minimax polynomial. The minimax polynomial is obtained by an iterative method called the Remez exchange algorithm, in which Gröbner bases computation is employed. We present an effective way to compute the minimax polynomial using Gröbner bases.

Keywords

Estimation problem, Function approximation, Minimax polynomial, Remez exchange algorithm, Gröbner basis, Speech direction estimation problem

1. Intorduction

In this paper, we discuss solving estimation problems with a function approximation method using the minimax polynomial and computer algebra.

Function approximation is the technique to approximate functions. It is used to make a sequence of polynomials for proving the density of function space [1] or to regard a function as a polynomial for evaluation [2]. Various methods for function approximation have been proposed, such as the Maclaulin expansion or the least squares method [2]. Here, as one of them, we present the minimax approximation and Remez exchange algorithm. The minimax approximation is the approximation using the polynomial which meets the property that the maximum value of the difference between the given function and the derived polynomial is the smallest of all polynomials in a given domain. The polynomial satisfying such a property is called the minimax polynomial. Since minimax polynomials are polynomials, one can use algebraic computation. On the other hand, the interpolation method [2], which is frequently used in computer algebra, estimates the polynomial based on discrete points and values on the original function. Compared with the interpolation method, the minimax approximation is better for errors, that is, the maximum value of errors of the minimax polynomial is less than that of the polynomial obtained by the interpolation method in many cases. In computing the minimax polynomial, an iteration method called the Remez exchange algorithm is used.

Estimation problems are the problems of estimating unknown information using already known information. Solving estimation problems is important in developing devices since objects that are measurable are limited. To solve estimation problems, one uses numerical methods such as the gradient method [3] or the genetic Algorithms [4]. However, the gradient method may return a local solution depending on initial values since it uses local convergence properties, and the genetic algorithm has some disadvantages in that it sometimes solves the estimation problem not properly due to the phenomena called initial convergence and hitchhiking. On the other hand, the estimation method using minimax

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

[⋆]This work was partially supported by JKA and its promotion funds from KEIRIN RACE.

[*]Corresponding author.

s2320136@u.tsukuba.ac.jp (K. Kuramochi); terui@math.tsukuba.ac.jp (A. Terui); mikawa@slis.tsukuba.ac.jp (M. Mikawa)
https://researchmap.jp/aterui (A. Terui); https://mikawalab.org/ (M. Mikawa)
0000-0003-0846-3643 (A. Terui); 0000-0002-2193-3198 (M. Mikawa)

approximation together with Gröbner bases [5] computation may avoid these phenomena, for this method evaluates values globally.

In this paper, we propose a method for solving estimation problems using the minimax polynomial and Gröbner bases, as follows. First, for a given mathematical model (function) of the estimation problem, we calculate the minimax polynomial which approximates the given function. Then, we make a system of polynomial equations and solve it with Gröbner bases computation for obtaining the solution of the estimation problem. Furthermore, we apply the proposed method to the speech direction estimation problem, which is an estimation of the direction of a speaker using a microphone array. For the speech direction estimation problem, a method using the Genetic Algorithm has been proposed [6]. We show that we can effectively use the minimax approximation and Gröbner bases for finding global solutions to the speech direction estimation problem.

The paper is organized as follows. In Section 2, we review the definition of the minimax polynomial and the Remez exchange algorithm. In Section 3, after defining the estimation problem more minutely, we propose a solution for this using the minimax approximation and Gröbner bases. In Section 4, we introduce the speech direction estimation problem, which is the task we are working on, and explain the reason why the method in this paper can be used. In Section 5, we conclude and pick up some challenges we are facing and tasks to improve our tasks.

2. Preliminaries

Let K be a field. The notation $K[\boldsymbol{x}]$ or $K[x_1, \dots, x_n]$ stands for the ring of polynomials over K in $x_1, \dots, x_n$. For a function f, $\|f\|_\infty$ denotes the infinity norm of f.

2.1. Minimax Approximation

In the following, let $[a, b]$ be a closed interval and f be a continuous function on $[a, b]$.

Definition 2.1 (Minimax polynomial). For the function f in above, a polynomial $P \in K[x]$ of degree k which minimizes

$$\|f - P\|_\infty = \max_{a \le x \le b} |f(x) - P(x)|,$$

is called *the k-th minimax polynomial of f.*

Definition 2.1 says that, if $P \in K[x]$ is the k-th minimax polynomial of f, the inequality

$$\max_{a \le x \le b} |f(x) - P(x)| \le \max_{a \le x \le b} |f(x) - Q(x)|,$$

follows for any $Q \in K[x]$ with $\deg Q = k$. In other words, the k-th minimax polynomial is the best polynomial in terms of error. Thus, one should use the minimax polynomial if one wants to approximate the given function by polynomials with minimizing errors.

The following theorem [2] tells us that that that value has a minimum value.

Theorem 2.1. *If the function f is continuous, the f has a unique k-th minimax polynomial for any $k \in \mathbb{Z}_{\ge 0}$. Furthermore, we can construct the k-th degree minimax polynomial by Algorithm 1.*

For details of Algorithm 1 such as uniqueness and termination, see [7]. Note that Algorithm 1 needs to compute the solution of a system of linear equations and the extremum points of continuous functions. (We have implemented Algorithm 1 using a computer algebra system Risa/Asir [8] with the library os_muldif [9] [1].)

The method to approximate functions using the Remez exchange algorithm is called *minimax approximation.* Note that the maximum error of $f - P$ becomes smaller as the degree of minimax

[1]Solving a system of linear equations is performed by Risa/Asir itself and the function"os_md.fmmx()" in the os_muldif library is used for computing extremum points with the extrema.

Algorithm 1 Remez exchange algorithm [2]

```
Require: a continuous function $f$, a closed interval $[a, b]$, a degree of a polynomial$k$
Ensure: a minimax polynomial $P \in K[x]$ with degree $k$
1: $E_{\max} := \infty,\ E_{\min} := 0$
2: $I := [x_0, x_1, \dots, x_k, x_{k+1}]$ : $k + 2$ initial interpolate points
3: while $E_{\max} >> E_{\min}$ do
4:     $[a_0, a_1, \dots, a_k, E]$ : the solution of $\sum_{j=0}^{k} a_j(x_i)^j + (-1)^i E = f(x_i)\ \ (i = 0, 1, \dots, k, k+1)$
5:     $P := \sum_{j=0}^{k} a_j x^j$
6:     $I := [x_0, x_1, \dots, x_k, x_{k+1}]$ : the $k + 2$ extremum points of $E(x) := f(x) - P(x)$
7:     $E_{\max} := \max\{|E(x_0)|, |E(x_1)|, \dots, |E(x_{k+1})|\},\ E_{\min} := \min\{|E(x_0)|, |E(x_1)|, \dots, |E(x_{k+1})|\}$
8: end while
9: return $P$
```

polynomial k becomes larger and the interval $[a, b]$ more restricted. The method in this paper can be used if functions are continuous, their range is bounded and the variable to solve is in a bounded interval. If the variable to solve is in a bounded interval that is not closed, we can construct the minimax polynomial over a closed one containing it.

Note that the minimax polynomial P of f has the following property: the error function $f - P$ has $\deg P + 2$ maximum value with alternate change of singers in $[a, b]$. The value of $|f(x) - P(x)|$ rapidly becomes larger as the value x separates from $[a, b]$.

3. The estimation problem

In this section, we introduce the method to solve estimation problems using minimax approximation and computer algebra.

Let $\boldsymbol{u} = (u_1, \dots, u_n)$ be measurable parameters and $\boldsymbol{v} = (v_1, \dots, v_t)$ be immeasurable parameters, and x be the parameter to be estimated. Assume that a mathematical model describing phenomena is given as

$$f(\boldsymbol{u}, x) + \sum_{i,j,k} g_i(\boldsymbol{u}) h_j(v_k) = 0, \tag{1}$$

where f is a continuous function and x is bounded. Since $\boldsymbol{u}$ are measurable, one can consider $f(\boldsymbol{u}, x)$ as a function only in x, $g_i(\boldsymbol{u})$ as a constant by substituting $\boldsymbol{u}$ and $h_i(v_j)$ as a new immeasurable parameter by replacing it with a new variable properly if necessary. Thus, f can be approximated by the minimax polynomial and the equation is transformed into the form of

$$P(x) + \sum_{i} Q_i(\boldsymbol{v}) = 0, \tag{2}$$

where P and Q_i represents the polynomials in x and $\boldsymbol{v}$, respectively. As eq. (2) is a multivariate polynomial equation, a system of polynomial equations can be generated by substituting measured values into measurable parameters. Thus, Gröbner bases and the Elimination Theorem [5] can be used to solve the system.

Note that the minimax approximation is the best way to approximate a function by polynomials in terms of errors. However, under some conditions, the minimax approximation cannot be performed. To be able to apply the approximation, the mathematical model with the estimated parameter must be continuous, have no other immeasurable parameters, and the estimated parameter must be in a bounded interval. The more restricted the interval is, the better the accuracies of the minimax approximations are. Thus this method is suitable for estimation problems in that the range of the solution of the estimated parameters is bounded.

4. An example by speech direction estimation problem

Speech direction estimation problem [6] is the problem of estimating the orientation of the speaker's face. This problem derives from robotics. Nowadays robots have been playing important roles in various fields and some research on robots cooperating with each other is underway [10, 11]. For such tasks, how to select one robot from more than one is one of the themes when humans select and command one to do some tasks. Thus, we are trying to select one robot by calling like humans saying "excuse me" or "hey." We assume a situation in a room like we are calling one robot waiter from many in a restaurant. Thus naming each robot is difficult.

We suppose that we know the coordinates of a user and robots, for the microphone arrays we are assuming have a function to compute the locations of themselves and the directions of arrival (DoA) and we can compute the user's location using DoA. For details, see [6]. To estimate the direction of the speaker's face orientation, we use a mathematical model called the voice spread model. The voice spread model is a formula to compute logical sound pressure levels recorded by a particular microphone array. We consider the situation in which there is a speaker and microphone array mic i. To construct a voice spread model, we need to consider two attenuation effects, distance attenuation and angle attenuation.

Distance attenuation is the effect that the longer the distance between a sound source and a sound receiving point is, the smaller the sound pressure level of the receiving point is. Supposing a sound source to be a point, the sound pressure level is in inverse proportion to the square of the distance.

Angle attenuation is the effect that the sound pressure level is the strongest in the front direction of a mouth and it gets weaker when separate from the direction. There are various kinds of research on angle attenuations [12, 13, 14] and we adopt Monson's [15].

Let Σ_W be the world coordinate system, Σ_S be the local coordinate system whose center is the user and the x-axis is the front direction of the user's face. Furthermore, let θ be the angle formed by the x-axis of Σ_W and Σ_S. Note that θ is the very estimated parameter. Then, the coordinate of mic i is expressed as

$$^{S}p_i = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} ({}^{W}p_i - {}^{W}p_S).$$

The parameters ${}^{W}p_i$ and ${}^{W}p_S$ stand for the coordinate of mic i and the user in Σ_W, respectively. Then, the theoretical sound pressure level $\hat{L}_i$ recorded by mic i is described as

$$\hat{L}_i = L - 10\log_{10} \|{}^{S}p_i\|^2 - a\left(1 - \frac{1+\cos(\varphi_i)}{2}\right).$$

Note that parameter L is the sound pressure level of the point that is 1 [m] away from the speaker and in the direction of the speaker's mouth. A parameter a is the size of angle attenuation and ϕ_i represents the azimuth angle to mic i in Σ_S. Note that the parameters L and a are immeasurable and ϕ_i is measurable.

Let Σ_W be the world coordinate system, Σ_S be the local coordinate system whose center is the user and x-axis is the front direction of the user's face. Also let θ be the angle formed by x-axises of Σ_W and Σ_S. Note that θ is the very estimated parameter. The coordinate of mic i's microphone array is expressed as

$$^{S}p_i = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} ({}^{W}p_i - {}^{W}p_S)$$

The parameters ${}^{W}p_i$ and ${}^{W}p_S$ stands for the coordinate of mic i and the user in Σ_W respectively. Then, the theoritical sound pressure level $\hat{L}_i$ recorded by mic i is described as

$$\hat{L}_i = L - 10\log_{10} \|{}^{S}p_i\|^2 - a\left(1 - \frac{1+\cos(\varphi_i)}{2}\right)$$

Note that parameter L is the sound pressure level of the point that is 1 [m] away from the speaker and in the direction of the speaker's mouse. A parameter a is the size of angle attenuation and ϕ_i represents the azimuth angle to mic i in Σ_S. Make sure that the parameters L and a are immeasurable and ϕ_i is measurable.

Since ${}^{W}p_i$, ${}^{W}p_S$, and φ_i are measurable parameters, one can substitute measured values into these. Thus, the term $\left(1 - \dfrac{1+\cos(\varphi_i)}{2}\right)$ can be alternated by a measured parameter m_i and $\hat{L}_i$ can be substituted by a measured value. Thus, the equation above can be transformed into the form of

$$\hat{L}_i - L + 10 \log_{10} \|{}^{S}p_i\|^2 + am_i = 0.$$

Since the term $\log_{10} \|{}^{S}p_i\|^2$ can be regarded as a function with the variable θ only and θ is in a bounded interval $(-\pi, \pi]$, we can approximate that term by the minimax polynomial $P_i(\theta)$. As a result, we have

$$\hat{L}_i - L + 10P_i(\theta) + am_i = 0.$$

The equation above is a multivariate polynomial in L, θ, a for any number i. To find $\theta\,(-\pi < \theta \leq \pi)$, we need to measure parameters ${}^{W}p_i$, ${}^{W}p_S$, φ_i, $\hat{L}_i$ for given $i \geq 3$, thus we compute the Gröbner bases and make the system of equations a triangular form.

5. Concluding remarks

In this paper, we have proposed a method for estimation problems using the minimax approximation and Gröbner bases computation. As an application, we have shown that the method can be used for the speech direction estimation problem.

To check the accuracy of the proposed method, experiments should be carried out in simulations and actual environments. For simulations of speech direction estimation problems, Python language has a library called Pyroomacoustics [16]. It is a software library to develop and test algorithms for voice processing, and can simulate an environment in a room and one can adjust the properties of the environment such as the location and the directivity of a sound source and a microphone, temperature, humidity, dimension, size and shape of a room and so on.

Furthermore, there exist the minimax rational functions and algorithms for computing ones. The minimax rational function of f is a rational function $R_{m,k}$ which minimizes

$$\max_{a \leq x \leq b} |f(x) - R_{m,k}(x)| = \max_{a \leq x \leq b} \left| f(x) - \frac{\sum_{j=0}^{m} a_j x^j}{\sum_{j=0}^{k} b_j x^j} \right|,$$

for given $m, k \in \mathbb{Z}_{\geq 1}$. Compared with the minimax polynomials, the minimax rational functions tend to have less errors. However, we failed to compute one properly due to errors in the calculation of improper integrals. We need to compute Chebyshev expansion [2] of f to construct a minimax rational function and to compute improper integrals. In our experiment, we failed to compute improper integrals properly due to divergence of integrands. Thus, the method to compute improper integrals with fewer errors should be investigated.

To solve estimation problems independent of the values of measured parameters, we need to approximate multivariate functions with multivariate polynomials or rational functions. Cody [17] says almost no algorithm for approximating those with minimax polynomials (or rational functions) exists. However, Luke [18] has made approximations of a variety of functions in mathematical physics using hypergeometric functions. Loeb [19] reports a linear algorithm to create rational approximations over a discrete point set. Fox, Goldstein and Lastman [20] also have proposed an algorithm for rational approximation on finite point sets.

The problem with which this method can be used is very limited since mathematical models that satisfy equation (1) and estimated parameters to be bounded are very restrictive. However, direction estimation problems are good because parameters of direction are usually bounded on $(-\pi, \pi]$ or $[0, 2\pi)$. To see the efficacy or effectiveness of this method, we should seek estimation problems that can be solved by this method.

References

[1] H. M. Stone, The generalized weierstrass approximation theorem, Mathematics Magazine 21 (1948) 167–184. doi:10.2307/3029750.

[2] H. M. Antia, Numerical methods for scientists and engineers, second ed., Basel; Boston: Birkhäuser, 2002. doi:10.1007/978-93-86279-52-1.

[3] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer, 1997. doi:10.1007/978-1-4612-0663-7.

[4] M. Mitchell, An Introduction to Genetic Algorithms, The MIT Press, 1998. doi:10.7551/mitpress/3927.001.0001.

[5] D. Cox, J. Little, D. O'shea, M. Sweedler, Ideals, varieties, and algorithms: An Intorduction to Computational Algebraic Geometry and Commutative Algebra, fourth ed., Springer, 2015. doi:10.1007/978-3-319-16721-3.

[6] T. Kato, Individual Selection Method for Multi-Robot based on Speech Direction Estimation (in Japanese), Master's thesis, Institute of Library, Information and Media Science, University of Tsukuba, 2024. 82 pages.

[7] N. Daili, A. Guesmia, Remez algorithm applied to the best uniform polynomial approximations, General Mathematics Notes 17 (2013) 16–31.

[8] M. Noro, T. Takeshima, Risa/Asir — A Computer Algebra System, in: ISSAC '92: Papers from the International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery, New York, NY, USA, 1992, pp. 387–396. doi:10.1145/143242.143362.

[9] T. Oshima, os_muldif: a library for computer algebra Risa/Asir, 2007–2024. URL: https://www.ms.u-tokyo.ac.jp/~oshima/muldif/os_muldifeg.pdf, accessed 2024-06-05.

[10] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Magazine 29 (2008) 9–19. doi:10.1609/aimag.v29i1.2082.

[11] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, P. Beardsley, Image and animation display with multiple mobile robots, The International Journal of Robotics Research 132 (2012) 753–773. doi:10.1177/0278364912442095.

[12] W. T. Chu, A. Warnock, Detailed directivity of sound fields around human talkers, Technical Report RR-104, National Research Council of Canada, 2002. doi:10.4224/20378930.

[13] G. A. Studebaker, Directivity of the human vocal source in the horizontal plane, Ear and hearing 6 (1985) 315–319. doi:10.1097/00003446-198511000-00007.

[14] D. Cebrera, P. J. Davis, A. Connolly, Long-term horizontal vocal directivity of opera singers: Effects of singing projection and acoustic environment, Journal of Voice 25 (2011) e291–e303. doi:10.1016/j.jvoice.2010.03.001.

[15] B. B. Monson, E. J. Hunter, B. H. Story, Horizontal directivity of low-and high-frequency energy in speech and singing, The Journal of the Acoustical Society of America 132 (2012) 433–411. doi:10.1121/1.4725963.

[16] R. Scheibler, E. Bezzam, I. Dokmanić, Pyroomacoustics: A python package for audio room simulation and array processing algorithms, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 351–355. doi:10.1109/ICASSP.2018.8461310.

[17] W. J. Cody, A survey of practical rational and polynomial approximation of functions, SIAM Review 12 (1970) 400–423. doi:10.1137/1012082.

[18] Y. L. Luke, The special functions and their approximations, volume 1, Academic Press, 1969.

[19] H. L. Loeb, Algorithms for chebyshev approximation using the ratio of linear forms, Journal of the Society for Industrial and Applied Mathematics 8 (1960) 458–465. doi:10.1137/0108031.

[20] P. Fox, A. Goldstein, G. Lastman, Rational approximations on finite point sets, in: Approximation of Functions (Proc. Sympos. General Motors Res. Lab., 1964), 1965, p. 57.

First-Order Theorem Proving with Power Maps in Semigroups*

Yi Lin[1,†], Ranganathan Padmanabhan[2,†] and Yang Zhang[3,*,†]

[1]*Department of Mathematics, The Ohio State University, Ohio, USA*
[2]*Department of Mathematics, University of Manitoba, Winnipeg, MB, Canada*
[3]*Department of Mathematics, University of Manitoba, Winnipeg, MB, Canada*

Abstract
This paper deals with automated deduction techniques to prove and generalize some well-known theorems in group theory that involve power maps, i.e., functions of the form $f(x) = x^n$. Here, the main obstacle is that if n is interpreted as an integer variable, then these results are not expressible in first-order logic with equality. The strategy followed here is to look at the classical proofs, involving the integer variable n, and see what specific first-order properties of power maps that are needed in the proofs. Then we implement these first-order properties of power maps in a theorem prover Prover9 and demonstrate that a well-designed reformulation makes specific mathematical theories accessible to the modern first-order theorem-proving software, allowing even for generalizations of the classical results.

Keywords
Semigroup, Prover9, Power maps

1. Introduction

The theory of groups and that of semigroups are very closely related. In fact, every group is a cancellation semigroup and, by a classical theorem of O. Ore [14], every cancellation semigroup satisfying some nontrivial identity, say $f(x, y) = g(x, y)$, is embeddable in a group. Also, there are several examples of identities $f = g$ which are preserved during this process of expansion. The most well-known example of a semigroup law that is transferable to groups is, of course, the commutative law. A.I. Mal'cev [8] and B.H. Neumann [12] have shown independently that nilpotent semigroup laws are transferable. However, it is also known that not all semigroup laws are preserved under the Ore extension. This raises the important question of finding more (and possibly all) transferable semigroup laws. This problem was raised by G.M. Bergman ([1, 2]).

There are several transferability theorems in semigroups that involve power maps $f(x) = x^n$. For example, it is known that every cancellation semigroup satisfying $x^n \cdot y^n = y^n \cdot x^n$ can be embedded in a group satisfying the same identity. Such statements belong to first-order logic with equality and hence provable, in principle, by any first-order theorem prover. However, because of the presence of an arbitrary integer parameter n in the exponent, they are outside the scope of any first-order theorem prover. In particular, one cannot use such an automated reasoning system to prove theorems involving power maps. Here we focus just on the needed properties of power map $f(x) = x^n$ and show how one can easily avoid having to reason explicitly with integer exponents. Implementing these new equational rules of power maps, we show how a theorem prover can be a handy tool for quickly proving or confirming the truth of such theorems involving power maps without explicitly mentioning the integer variable n.

Following Macedonska [7], a positive semigroup law is said to be transferable if being satisfied in a cancellative semigroup S it must be satisfied in SS^{-1}, the group of right quotients of S. The most well-

SCSS 2024: The 10th International Symposium on Symbolic Computation in Software Science, August 28–30, Tokyo, Japan
*Corresponding author.
†These authors contributed equally.
yilinmaths@gmail.com (Y. Lin); Ranganathan.Padmanabhan@umanitoba.ca (R. Padmanabhan); Yang.Zhang@umanitoba.ca (Y. Zhang)

known example of a transferable law is, of course, the commutative law. A. I. Malcev, B. H. Neumann and others ([8, 12]) have shown that nilpotent identities are transferable. Macedonska [7] has proved the transferablity of several two-variable semigroup laws. These identities are defined by using power maps $f(x) = x^n$ in semigroups.

Here we will replace the power map by power-like functions and prove their transferability. The transferability of identities is first order problem but first-order theorem provers cannot handle power-maps because of the presence of an integer variable "n". Here we demonstrate that computers can prove these semigroup implications, thus generalizing what is known classically.

A motivating example: It is well-known that in groups, the commutators $[x,\ y]$ can be expressed in a product of three squares, that is,

$$[x,\ y] = x^{-1}y^{-1}xy = (x^{-1})^2 \cdot (xy^{-1})^2 \cdot y^2,$$

and hence $x^2y = yx^2$ implies $[x,\ y]$ is central, which is equivalent to the semigroup implication

$$x^2y = yx^2 \implies xyzyx = yxzxy.$$

Definition 1.1. A cancellation semigroup $(G, \cdot)$ is a semigroup with the two-sided cancellative properties, i.e., for all $x, y, z \in G$, the following are true:

(i) $x \cdot y = x \cdot z$ implies $y = z$,
(ii) $x \cdot y = z \cdot y$ implies $x = z$.

Some properties of cancellation semigroup can be found in, for example, [4, 5, 13, 7, 16, 17]. Here we show that the above implication is valid in cancellation semigroups:

$$\begin{aligned} y \cdot yxzxy \cdot x &= y^2xzxyx \\ &= xzy^2xyx \\ &= xzy \cdot (yx)^2 \\ &= (yx)^2xzy \\ &= yxyx^2zy \\ &= yxyzyx^2 \\ &= y \cdot xyzyx \cdot x. \end{aligned}$$

Canceling y and x, we have $xyzyx = yxzxy$.

Next we present the proof by using Prover9 [9].

```
%% INPUT file
%% In groups, squares are central ==> commutators are also central
(x * y) * z = x * (y * z).
x * e = x.
x * x' = e.
x * y = (y * x) * [x, y]. %% commutators defined
x * (y * y) = (y * y) * x. %% squares are central
%% goal to prove that commutators are central
x * [y, z]  =  [y, z] * x.
=============== PROOF ================================
1 x * [y,z] = [y,z] * x # label(non_clause) # label(goal).
2 (x * y) * z = x * (y * z).  [].
3 x * e = x.  [].
4 x * x' = e.  [].
5 x * y = (y * x) * [x,y].  [].
```

```
6 x * (y * [y,x]) = y * x.  [5,2].
7 x * (y * y) = (y * y) * x.  [].
8 x * (y * y) = y * (y * x).  [7,2].
9 [c2,c3] * c1 != c1 * [c2,c3].  [1].
10 x * (e * y) = x * y.  [3,2].
11 x * (x' * y) = e * y.  [4,2].
12 x * (y * (x * y)') = e.  [4,2].
13 x * (y * ([y,x] * z)) = y * (x * z).  [6,2,2,2].
14 x * (y * (z * [z,x * y])) = z * (x * y).  [6,2].
15 x * (y * (z * [y * z,x])) = y * (z * x).  [2,6,2].
16 x * (y * (y * z)) = y * (y * (x * z)).  [8,2,2,2,2].
20 e * x = x.  [3,8,3,10].
21 x * (x' * y) = y.  [11,20].
29 x'' = x.  [4,21,3].
31 x * [x,y'] = y * (x * y').  [6,21].
33 x * (y * (x' * x')) = x' * y.  [8,21].
35 x' * x = e.  [29,4].
36 x' * (x * y) = y.  [29,21].
40 x * (y * [x,y]') = y * x.  [4,13,3].
41 x * (y * (z * [z,[x,y]])) = y * (x * (z * [x,y])).[6,13].
43 x * (y * (z * (z * [y,x]))) = y * (x * (z * z)). [8,13].
55 x' * (y * x) = y * [y,x].  [6,36].
58 x * (y * x)' = y'.  [12,36,3].
66 x * (y * [y,z' * x]) = z * (y * (z' * x)).  [14,21].
80 x*(y *(z*(u*[y*(z*u),x])))=y*(z*(u*x)).[2,15,2,2].
98 (x * y)' = y' * x'.  [58,36].
112 x * (y * (z * (z * u))) = z * (z * (x * (y * u))).[16,2,2].
115 x * (x * (y * x')) = y * x.  [4,16,3].
127 x * (y * (x' * (x' * z))) = x' * (y * z).  [16,21].
128 x * (x * (y * (x' * z))) = y * (x * z).  [21,16].
189 x * (x * (y * (z * x'))) = y * (z * x).  [2,115,2].
190 x' * (y * x) = x * (y * x').  [115,21,29].
206 x * [x,y] = y * (x * y').  [55,190].
326 x' * (y * (x * z)) = x * (y * (x' * z)).  [190,2,2,2,2].
467 x * (y' * [y,x]) = y' * x.  [33,13,33].
481 x * [y,x]' = y * (x * y').  [40,36,190].
512 x * (y * (x' * y')) = [x',y'].  [31,21].
516 [x',y'] = [x,y'].  [31,36,326,512].
543 x * (y * (x' * y')) = [x,y'].  [512,516].
547 [x',y] = [x,y'].  [206,21,543].
551 [x,y'] = [x,y].  [206,36,326,543].
552 x' * (y * [y,x]) = y * x'.  [206,36].
569 x * [y,x] = y * (x * y').
    [206,190,98,29,2,36,98,29,2,36].
584 [x',y] = [x,y].  [547,551].
585 x * (y * (x' * y')) = [x,y].  [543,551].
592 x' * (y * ([y,x] * z)) = y * (x' * z).  [551,13].
605 x * (y * [x,y]) = y * x.  [569,8,8,115].
606 [x,y] = [y,x].  [569,21,585,551].
754 x * (y' * [x,y]) = y' * x.  [551,605].
1004 [x,y]' = [y,x].  [481,21,585,551].
1049 x * [y,[x,y]] = x.  [21,41,584,754,21].
```

```
1095 [x,[y,x]] = e.  [1049,21,4,584].
1157 [x,y' * (y' * x')] = e.
     [35,43,98,98,2,98,98,2,606,2,2,326,36,36,21].
1383 [x,x * (y * y)] = e.  [1157,551,98,98,29,29,29,2].
1387 [x,y * (y * x)] = e.  [8,1383].
1400 [x * y,x' * y] = e.  [21,1387,606].
1534 [x * (y * z),x * (y' * z)] = e.  [13,1400,592].
1861 x * (y * (x' * (y' * z))) = [x,y] * z.  [585,2,2,2].
1863 x * (y * (z * (x' * (z' * y')))) = [x,y * z].[2,585,98].
1874 x * ([y,z] * (x' * [z,y])) = [x,[y,z]].  [1004,585].
1903 [x,y' * x] = [y,x].  [66,21,190,585,584].
2118 [x,y * x] = [y,x].  [29,1903,584].
2121 [x,x * y] = [x,y].  [36,1903,606,2118].
2148 [x' * y,[x,y]] = e.  [1903,1095].
2195 [x * y,[x,y]] = e.  [2118,1095].
2365 [x * y,y * x] = e.  [2195,2121,2,605].
2384 [x * y,x' * y'] = e.  [2365,551,98].
2592 [x,y] * (y' * x) = x * y'.
     [2148,467,98,29,3,98,29,2,552].
2674 x' * (y' * (z * (x * (z' * y)))) = [x,z * y].
     [2384,80,98,3,2,2,326,98,1863].
3912 x*(y*(z*(x'*(x'*u))))=x'*(y*(z*u)).[112,21].
4517 [x,[y,z]] * ([z,y] * x) = x * [z,y]. [1004,2592,1004].
8121 x*(y*(x'*(z*(x'*z'))))=x'*(y*[x,z]). [206,127,584].
8175 x'*(y*(z*(x*u)))=x*(y*(z*(x'*u))). [128,127,29].
8179 x'*(y*(z*(u*x)))=x*(y*(z*(u*x'))). [189,127,29].
8223 [x,y * z] = [x,z * y].  [2674,8179,8175,1863].
8383 [x * (y * z),y' * (z * x)] = e.  [8223,1534,2].
13972 [x,[y,z]] = e.  [8383,585,98,98,2,98,98,29,2,2,2,8179,
                       2,2,3912,8121,2,2,1861,1874].
14283 [x,y] * z = z * [x,y].  [4517,13972,20].
14284 $F.  [14283,9].
==================== end of proof ========================
```

Coda: In the human proof above, we already "knew" that commutators are expressible as a product of squares in the group and hence the human proof was almost trivial. But in the above machine proof of the same fact, the Prover9 is not even "aware" of the fact that commutators are products of squares. Still, the software did prove the centrality of commutators as explicitly shown in line #14283 above (proved with the Knuth-Bendix option). Dr. William McCune, the author of Prover9, has done a great job.

In this paper, we first consider the implication $(xy)^n = (yx)^n$ in cancellation semigroups. In Section 2, we prove that this implication is equivalent to the identity $xy^n = y^n x$ in all cancellation semigroups by replacing the power-map x^n by a weaker power-like function $f(x)$. Furthermore, we discuss a general extension. In Section 3, we prove that $xy^n = y^n x$ is transferable.

2. Power map properties

We first list some properties of power maps. We refer the readers to [11, 15] for more details.

Lemma 2.1. *Let $(S;\cdot)$ be a cancellation semigroup and let $f : S \to S$ be the usual power map $f(x) = x^n$, for some $n > 1$. Assume that $f(x \cdot y) = f(y \cdot x)$. Then the function $f(x)$ satisfies the following:*

(1) $x \cdot f(x) = f(x) \cdot x$.
(2) $x \cdot f(y \cdot x) = f(x \cdot y) \cdot x$.

(3) x and $f(f(x))$ commute.
(4) If x and y commute then $f(x \cdot y) = f(x) \cdot f(y)$.
(5) x and $f(y \cdot x)$ commute.
(6) x and $f(f(x) \cdot y)$ commute.

Proof. (1) is obvious since both sides are equal to x^{n+1}.
(2) $x \cdot f(y \cdot x) = x \cdot (y \cdot x)^n = (x \cdot y)^n \cdot x = f(x \cdot y) \cdot x$.
(3) follows that $f(f(x))$ is just a power of x and hence commutes with x.
(4) is obvious thanks to associativity and commutativity.
(5) $x \cdot f(y \cdot x) = f(x \cdot y) \cdot x = f(y \cdot x) \cdot x$ since $f(x \cdot y) = f(y \cdot x)$.
(6)

$$\begin{aligned} f(f(x) \cdot y) \cdot x &= f(g(x) \cdot x \cdot y) \cdot x && \text{where } g(x) = x^{n-1} \\ &= f(x \cdot y \cdot g(x)) \cdot x && \text{since } f(x \cdot y) = f(y \cdot x) \\ &= x \cdot f(y \cdot g(x) \cdot x) && \text{by (2) above} \\ &= x \cdot f(y \cdot f(x)) && \text{since } f(x) = g(x) \cdot x \\ &= x \cdot f(f(x) \cdot y) && \text{since } f(x \cdot y) = f(y \cdot x) \end{aligned}$$

Hence the two elements x and $f(f(x) \cdot y)$ commute. In particular, the two terms $y \cdot x$ and $f(f(y \cdot x) \cdot x)$ commute. □

Following the terminology of [11, 15], we call the unary functions $f(x)$ satisfying first-order properties (1) to (4) of Lemma 2.1 as *power-like functions.*

Theorem 2.2. *Let S be a cancellation semigroup and let $f : S \to S$ be a power-like function and assume that $f(x \cdot y) = f(y \cdot x)$. Then $f(x)$ is a central element in S.*

We can prove this theorem by using our method and Prover9. Here we list the a few lines of output of Prover9 which include the conditions and the final result.

```
========================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 916 was started by yangzhang
on yangzhangsimac2.ad.umanitoba.ca,
Tue Mar 19 12:26:28 2024
The command was "/Users/yangzhang/Desktop/Prover9-Mace4-v05B.app/
Contents/Resources/bin-mac-intel/prover9".
=========================== end of head ==========================
========================== end of input ==========================
======================== PROOF =================================
% -------- Comments from original proof --------
% Proof 1 at 0.84 (+ 0.02) seconds.
% Length of proof is 24.
% Level of proof is 6.
% Maximum clause weight is 29.
% Given clauses 117.

1 f(x) * y = y * f(x) # label(non_clause) # label(goal).  [goal].
2 (x * y) * z = x * (y * z).  [assumption].
3 x * y != x * z | y = z.  [assumption].
4 x * y != z * y | x = z.  [assumption].
5 f(x * y) = f(y * x).  [assumption].
6 f(x * y) * x = x * f(y * x).  [assumption].
7 f(x) * x = x * f(x).  [assumption].
```

```
8 f(f(x) * y) * x = x * f(f(x) * y).  [assumption].
9 x * y != y * x | f(x * y) = f(x) * f(y).  [assumption].
10 f(c1) * c2 != c2 * f(c1).  [deny(1)].
...........
...........
3645 f(f(x)) * x = x * f(f(x)).  [hyper(351,a,139,a),flip(a)].
3699 f(x)*(f(x)*(y*f(f(f(x)*y))))=f(x)*(y*(f(x)*f(f(f(x)*y)))).
[back_rewrite(395),rewrite([3645(8),2(8)])].
3700 $F.  [resolve(3699,a,67,a)].

======================= end of proof ==========================
```

Next, we give the human proof as follows:

$$\begin{array}{ll} f(x)\cdot y\cdot x\cdot f(f(y\cdot x)) & \\ = f(x)\cdot f(f(y\cdot x))\cdot y\cdot x & \text{since } u \text{ and } f(f(u)) \text{ commute} \\ = f(x\cdot f(y\cdot x))\cdot y\cdot x & \text{since } x \text{ and } f(y\cdot x) \text{ commute} \\ = f(f(\cdot x)\cdot x)\cdot (y\cdot x) & \text{since } f(x\cdot y)=f(y\cdot x) \\ = (y\cdot x)\cdot f(f(y\cdot x)\cdot x) & \text{since } y\cdot x \text{ and } f(f(y\cdot x)\cdot x) \text{ commute.} \\ = y\cdot x\cdot f(x\cdot f(y\cdot x)) & \text{since } f(x\cdot y)=f(y\cdot x) \\ = y\cdot x\cdot f(x)\cdot f(f(y\cdot x)) & \text{since } x \text{ and } f(y\cdot x) \text{ commute} \\ = y\cdot f(x)\cdot x\cdot f(f(y\cdot x)) & \text{since } x \text{ and } f(x) \text{ commute} \end{array}$$

Hence, we have

$$f(x)\cdot y\cdot x\cdot f(f(y\cdot x)) = y\cdot f(x)\cdot x\cdot f(f(y\cdot x)).$$

Finally, cancelling the common term $x\cdot f(f(y\cdot x))$ from the right sides, we get $f(x)\cdot y = y\cdot f(x)$.

Corollary 2.3. *In a cancellation semigroup $(S,\cdot)$, for $x, y \in S$ and $n \in \mathbb{Z}^+$, $(x\cdot y)^n = (y\cdot x)^n$ implies $x^n\cdot y = y\cdot x^n$.*

Proof. Simply take $f(x) = x^n$. The power map $f(x) = x^n$ satisfies all the six properties mentioned in Lemma 2.1 and hence the proof of Theorem 2.2 applies. Therefore, the n-th powers are central in the semigroup, i.e., $x^n\cdot y = y\cdot x^n$. □

Next we consider the following more general case.

Theorem 2.4. *In a cancellation semigroup S, if there exist $k \geq 2, n \in \mathbb{Z}^+$ such that $(a_1a_2\cdots a_k)^n = (a_k\cdots a_2a_1)^n$ for any $a_i \in S$ holds, then x^n is central in S for any $x \in S$.*

Proof. When $n = 1$, all the situations can be reduced to $k = 2$ or $k = 3$.

In case of $n = 1$ and $k = 2$, we have $a_1a_2 = a_2a_1$, and then S is commutative.

In case of $n = 1$ and $k = 3$, we have $a_1a_2a_3 = a_3a_2a_1$. Then, for any $x, y, z, u \in S$,

$$xyzu = uzxy = yzxu.$$

Cancelling u from the right sides, we obtain $xyz = yzx$. By the condition, $yzx = xzy$. Then $xyz = xzy$, and thus $yz = zy$. Hence S is commutative.

Next, we consider $n \geq 2$. Note that

$$\begin{aligned} x\cdot(xy^{k-1})^n &= x\cdot(x\cdot y\cdots y)^n \\ &= x\cdot(y\cdots y\cdot x)^n \\ &= x\cdot(y^{k-1}\cdot x)^n \\ &= (xy^{k-1})^n x, \end{aligned}$$

that is, x and $(xy^{k-1})^n$ commute, and thus $(xy^{k-1})^n = (y^{k-1}x)^n$.

Pick up $m \in \mathbb{Z}^+$ such that $mn \geq k-1$. Now we claim the following identity holds

$$x^n y[(xy^{k-1})^{mn}]^n = yx^n[(xy^{k-1})^{mn}]^n.$$

We will combine suitable x's and y's together and apply above commutative properties:

$$\begin{aligned}
x^n y[(xy^{k-1})^{mn}]^n &= x^n y[(xy^{k-1})^{mn-1}(xy^{k-2})y]^n \\
&= x^n[y(xy^{k-1})^{mn-1}(xy^{k-2})]^n y \\
&= x^n[y(xy^{k-1})^{mn-1}x\underbrace{y\cdots y}_{k-2}]^n y \\
&= x^n[\underbrace{y\cdots y}_{k-2}\cdot x\cdot y(xy^{k-1})^{mn-1}]^n y \\
&= x^n(y^{k-2}xy(xy^{k-1})^{mn-1})^n y \\
&= x^n[y^{k-2}xy\underbrace{(xy^{k-1})\cdot(xy^{k-1})\cdots(xy^{k-1})^{mn+2-k}}_{k-2}]^n y \\
&\quad (\text{ since } mn \geq k-1, mn+2-k \geq 1) \\
&= x^n[\underbrace{(xy^{k-1})^{mn+2-k}(xy^{k-1})\cdots(xy^{k-1})}_{k-2}\cdot xy\cdot y^{k-2}]^n y \\
&= x^n[(xy^{k-1})^{mn}]^n y \\
&= [x\cdot(xy^{k-1})^{mn}]^n y \\
&\quad (\text{since } x \text{ commutes with } (xy^{k-1})^n \text{ and } (xy^{k-1})^{mn}).
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
yx^n[(xy^{k-1})^{mc}]^n &= y[x\cdot(xy^{k-1})^{mn}]^n \text{ (since } x \text{ and } (xy^{k-1})^{mn} \text{ commute)} \\
&= y\cdot[x(xy^{k-1})^{mn-1}\cdot xy^{k-2}\cdot y]^n \\
&= [y\cdot x(xy^{k-1})^{mn-1}\cdot xy^{k-2}]^n y \\
&= [yx(xy^{k-1})^{mn-1}\cdot x\underbrace{y\cdots y}_{k-2}]^n y \\
&= \underbrace{y\cdots y}_{k-2}\cdot x\cdot yx(xy^{k-1})^{mn-1}]^n y \\
&= \underbrace{y\cdots y}_{k-2}\cdot(xy)\cdot x(xy^{k-1})^{mn-1}]^n y \\
&= [x(xy^{k-1})^{mn-1}\cdot(xy)\cdot\underbrace{y\cdots y}_{k-2}]^n y \\
&= [x(xy^{k-1})^{mn}]^n y
\end{aligned}$$

Therefore, cancelling $[(xy^{k-1})^{mc}]^n$ from the right sides of $x^n y[(xy^{k-1})^{mn}]^n = yx^n[(xy^{k-1})^{mc}]^n$, we have $x^n y = yx^n$. $\square$

3. Transferability of $(xy)^n = (yx)^n$

Let $(S, \cdot)$ be a cancellation semigroup satisfying the law $(xy)^n = (yx)^n$. Then, by Corollary 2.3, we know that all the powers of n are central in S, i.e., S satisfies the identity $x^n \cdot y = y \cdot x^n$. We now use the Ore principle resulting from this identity to construct the actual group of quotients. Thus we will have an explicit formula for the group multiplication. We show that this group multiplication satisfies the semigroup law $x^n \cdot y = y \cdot x^n$. This will prove the transferability of $x^n \cdot y = y \cdot x^n$ from S to its Ore group of right quotients SS^{-1}. Since $(xy)^n = (yx)^n$ and $x^n \cdot y = y \cdot x^n$ are equivalent in cancellation semigroups, we get that the semigroup law $(xy)^n = (yx)^n$ is also transferable.

Since the semigroup satisfies a non-trivial identity, it obviously satisfies the Ore left multiple principle (the property Mv in Ore [14]), the group of right quotients SS^{-1} exists. What is not obvious is that the group also satisfies the identity.

Theorem 3.1. *The semigroup law $(xy)^n = (yx)^n$ is transferable in a semigroup $(S, \cdot)$.*

Proof. We first define the multiplication and the equality of quotients. Let $\frac{a}{b}$ and $\frac{c}{d}$ be two right quotients in the group SS^{-1}. Thus the elements a, b, c, d are in S. We follow the idea of Ore and define the product and equality of two right quotients. Note that

$$\frac{a}{b} \cdot \frac{c}{d} = ab^{-1}cd^{-1} = ab^{n-1-n}cd^{n-1-n} = ab^{n-1}cd^{n-1}b^{-n}d^{-n} = \frac{ab^{n-1}cd^{n-1}}{b^nd^n}.$$

Thus we can define the product

$$\frac{a}{b} \cdot \frac{c}{d} = \frac{ab^{n-1}cd^{n-1}}{b^nd^n}.$$

Also, here

$$\tfrac{a}{b} \sim \tfrac{c}{d} \text{ if and only if } ab^{n-1}d = b^nc.$$

Hence, the identity is $\frac{a_0}{a_0}$ and the inverse $\left(\frac{a}{b}\right)^{-1} = \frac{b^n}{ab^{n-1}}$.

Next we define the embedding map as following: for any $a \in S$,

$$\phi : S \to SS^{-1}, \quad \phi(a) = \tfrac{aa_0}{a_0}. \quad \text{for some } a_0 \in S.$$

For any $a, b \in S$, we have

$$\phi(ab) = \frac{aba_0}{a_0} = \frac{aba_0^nb_0^n}{a_0^nb_0^n} = \frac{aa_0}{a_0}\frac{bb_0}{b_0} = \phi(a)\phi(b).$$

Hence ϕ is an isomorphism.

Now we prove that $\frac{a}{b} = \phi(a)\phi(b)^{-1}$ for any $a, b \in S$.

$$\phi(a)\phi(b)^{-1} = \frac{aa_0}{a_0}\left(\frac{bb_0}{b_0}\right)^{-1} = \frac{aa_0}{a_0}\frac{b_0^n}{bb_0^n} = \frac{aa_0^nb_0^n(bb_0^n)^{n-1}}{a_0^n(bb_0^n)^n}.$$

Thus

$$\begin{aligned}
\tfrac{a}{b} &= \tfrac{aa_0^nb_0^n(bb_0^n)^{n-1}}{a_0^n(bb_0^n)^n} \\
&\iff aa_0^nb_0^n(bb_0^n)^{n-1}b^n = ab^{n-1}a_0^n(bb_0^n)^n \\
&\iff aa_0^nb_0^n(bb_0^n)^{n-1}b^n = aa_0^nb^{n-1}(bb_0^n)(bb_0^n)^{n-1} \\
&\iff aa_0^nb_0^nb^n(bb_0^n)^{n-1} = aa_0^nb_0^nb^n(bb_0^n)^{n-1} \text{ (since } a_0^n, b^n, b_0^n \text{ are central).}
\end{aligned}$$

Hence $\frac{a}{b} = \phi(a)\phi(b)^{-1}$. Therefore, we have the following two formats for the group of quotients:

$$SS^{-1} = \left\{\frac{a}{b} \mid a, b \in S\right\} = \{\phi(a)\phi(b)^{-1} \mid a, b \in S\}.$$

For any $a, b \in S$, we can verify the following four identities:

$$\phi(a)^n\phi(b) = \phi(a^nb) = \phi(ba^n) = \phi(b)\phi(a)^n.$$

$$\phi(a)^n\phi(b)^{-1} = \phi(b)^{-1}\phi(b)\phi(a)^n\phi(b)^{-1} = \phi(b)^{-1}\phi(a)^n\phi(b)\phi(b)^{-1} = \phi(b)^{-1}\phi(a)^n.$$

$$\begin{aligned}
(\phi(a)^{-1})^n\phi(b) &= (\phi(a)^{-1})^n\phi(b)\phi(a)^n(\phi(a)^{-1})^n \\
&= (\phi(a)^{-1})^n\phi(a)^n\phi(b)(\phi(a)^{-1})^n \\
&= \phi(b)(\phi(a)^{-1})^n. \\
(\phi(a)^{-1})^n\phi(b)^{-1} &= \phi(b)^{-1}\phi(b)(\phi(a)^{-1})^n\phi(b)^{-1} \\
&= \phi(b)^{-1}(\phi(a)^{-1})^n\phi(b)\phi(b)^{-1} \\
&= \phi(b)^{-1}(\phi(a)^{-1})^n.
\end{aligned}$$

Since every element in SS^{-1} has the form $\phi(a)\phi(b)^{-1}$, we can conclude that both $\phi(a)^n$ and $(\phi(a)^{-1})^n$ are central in SS^{-1} for all $a \in S$.

Next, we prove that for any $a, b \in S$, $\left(\frac{a}{b}\right)^n = [\phi(a)\phi(b)^{-1}]^n$ is central in SS^{-1}. For any $g \in SS^{-1}$, we have

$$\begin{aligned}
[\phi(a)\phi(b)^{-1}]^n g &= [\phi(a)\phi(b)^{-1}][\phi(a)\phi(b)^{-1}]^{n-1}g \\
&= \phi(a)^n[\phi(a)^{-1}]^{n-1}\phi(b)^{-1}[\phi(a)\phi(b)^{-1}]^{n-1}g \\
&= \phi(ba^{n-1})^{-1}[\phi(a)\phi(b)^{-1}]^{n-1}g \cdot \phi(a)^n \\
&= [\phi(ba^{n-1})^{-1}]^n\phi(ba^{n-1})^{n-1} \cdot [\phi(a)\phi(b)^{-1}]^{n-1}g \cdot \phi(a)^n \\
&= [\phi(ba^{n-1})^{n-1}] \cdot [\phi(a)\phi(b)^{-1}]^{n-1}g \cdot \phi(a)^n[\phi(ba^{n-1})^{-1}]^n \\
&= (\phi[(ba^{n-1})^{n-1} \cdot a]\phi(b)^{-1}) \cdot [\phi(a)\phi(b)^{-1}]^{n-2}g[\phi(a)\phi(b)^{-1}] \\
&\quad \cdot[\phi(ba^{n-1})^{n-1}]^{-1} \\
&= (\phi[(ba^{n-1})^{n-1}a]\phi(b)^{-1}) \cdot [\phi(a)\phi(b)^{-1}]^{n-2}g[\phi(a)\phi(b)^{-1}] \\
&\quad \cdot[\phi(ba^{n-1})^{n-1}]^{-1} \\
&\qquad (\text{Let } a' = (ba^{n-1})^{n-1}a) \\
&= [\phi(a')\phi(b)^{-1}] \cdot [\phi(a)\phi(b)^{-1}]^{n-2}g[\phi(a)\phi(b)^{-1}][\phi(a')\phi(a)^{-1}]^{-1}.
\end{aligned}$$

Using the deduction above again, we have

$$\begin{aligned}
&(\phi[(b(a')^{n-1})^{n-1}a]\phi(b)^{-1} \cdot [\phi(a)\phi(b)^{-1}]^{n-3}g[\phi(a)\phi(b)^{-1}][\phi(a')\phi(a)^{-1}]^{-1} \\
&\cdot[\phi(a')\phi(b)^{-1}] \cdot [\phi(b(a'))^{n-1})^{n-1}]^{-1} \\
= \quad &(\phi[(b(a')^{n-1})^{n-1}a]\phi(b)^{-1} \cdot [\phi(a)\phi(b)^{-1}]^{n-3}g[\phi(a)\phi(b)^{-1}]^2 \\
&\cdot[\phi(b(a'))^{n-1})^{n-1}]^{-1}.
\end{aligned}$$

Define

$$a_1 = a,\ a_2 = a' = (ba^{n-1})^{n-1}a, \ldots, a_{k+1} = (ba_k^{n-1})^{n-1}a.$$

Then

$$\begin{aligned}
&[\phi(a)\phi(b)^{-1}]^n g \\
&= [\phi(a_1)\phi(b)^{-1}] \cdot [\phi(a)\phi(b)^{-1}]^{n-1} g \\
&= [\phi(a_2)\phi(b)^{-1}] \cdot [\phi(a)\phi(b)^{-1}]^{n-2} g[\phi(a)\phi(b)^{-1}] \cdot [\phi(a_2)\phi(a)^{-1}]^{-1} \\
&= \cdots \\
&= [\phi(a_k)\phi(b)^{-1}] \cdot [\phi(a)\phi(b)^{-1}]^{n-k} g[\phi(a)\phi(b)^{-1}]^{k-1} \cdot [\phi(a_2)\phi(a)^{-1}]^{-1} \\
&= \cdots \\
&= [\phi(a_n)\phi(b)^{-1}] \cdot g[\phi(a)\phi(b)^{-1}]^{n-1} \cdot [\phi(a_n)\phi(a)^{-1}]^{-1}
\end{aligned}$$

If $\phi(a_n)\phi(b)^{-1}$ is central, then we have

$$\begin{aligned}
&g[\phi(a)\phi(b)^{-1}]^{n-1} \cdot [\phi(a_n)\phi(a)^{-1}]^{-1}[\phi(a_n)\phi(b)^{-1}] \\
&= g[\phi(a)\phi(b)^{-1}]^{n-1} \cdot [\phi(a)\phi(b)^{-1}] \\
&= g[\phi(a)\phi(b)^{-1}]^{n}.
\end{aligned}$$

Hence, the proof is completed. Therefore, we will show that $\phi(a_n)\phi(b)^{-1}$ is central as follows.

Let $y_k = \phi(a_k)\phi(b)^{-1}$. Then

$$y_1 = \phi(a)\phi(b)^{-1},\ a_{k+1} = (ba_k^{n-1})^{n-1}a,$$

$$\phi(a_{k+1}) = \phi[(ba_k^{n-1})^{n-1}] \cdot \phi(a) = [\phi(b)\phi(a_k)^{n-1}]^{n-1} \cdot \phi(a).$$

Thus

$$\phi(a_{k+1})\phi(b)^{-1} = [\phi(b)\phi(a_k)^{n-1}]^{n-1} \cdot \phi(a)\phi(b)^{-1}.$$

That is,

$$\begin{aligned}
y_{k+1} &= [\phi(b)[y_k\phi(b)]^{n-1}]^{n-1} \cdot \phi(a)\phi(b)^{-1} \\
&= [[y_k\phi(b)]^n \cdot y_k^{-1}]^{n-1}\phi(a)\phi(b)^{-1} \\
&= [y_k\phi(b)]^{n(n-1)}(y_k^{-1})^n y_k\phi(a)\phi(b)^{-1}.
\end{aligned}$$

Continue the same deduction, and use y_{k-1} to represent y_k, we have

$$y_{k+1} = [y_k\phi(b)]^{n(n-1)}(y_k^{-1})^n[y_{k-1}\phi(b)]^{n(n-1)}(y_{k-1}^{-1})^n y_{k-1}[\phi(a)\phi(b)^{-1}]^2.$$

Thus

$$\begin{aligned}
y_n &= [y_{n-1}\phi(b)]^{n(n-1)}(y_{n-1}^{-1})^n y_{n-1}\phi(a)\phi(b)^{-1} \\
&= x_1^n x_2^n x_3^n x_4^n y_{n-2}[\phi(a)\phi(b)^{-1}]^2 \\
&= \cdots \\
&= x_1^n x_2^n \cdots x_{2n-2}^n y_1[\phi(a)\phi(b)^{-1}]^{n-1} \\
&= x_1^n x_2^n \cdots x_{2n-2}^n y_1[\phi(a)\phi(b)^{-1}]^{n},
\end{aligned}$$

where $x_i \in S, i = 1, \dots, 2n-2$. Since all $x^n, x \in S$ are central, y_n is central as desired. Therefore, $\left(\frac{a}{b}\right)^n$ is central, and thus the emdedding ϕ is perfect embedding. □

Acknowledgments

This project is partially supported by Mitacs Globalink Research Internship Canada NSERC.

References

[1] G. Bergman, *Hyperidentities of groups and semigroups.* Aequat. Math. 23 (1981), 55–65.

[2] G. Bergman, *Questions in Algebra.* Preprint, Berkeley, U.D. 1986.

[3] B. M. Green and J. R. Isbell, *Problems and Solutions: Solutions of Elementary Problems: E2259. Commuting powers in a group.* The American Mathematical Monthly, 78(8)(1971): 909–910.

[4] S. V. Ivanov, A. M. Storozhev, *On identities in groups of fractions of cancellative semigroups*, Proc. Amer. Math. Soc. 133 (2005), 1873–1879.

[5] J. Krempa and O. Macedonska, *On identities of cancellation semigroups*, Contemporary Mathematics, Vol 131, 1992

[6] F. W. Levi, *Notes on group theory. I, II,* J. Indian Math. Soc. 8 (1944), 1–9.

[7] O. Macedonska and P. Slanina. *On identities satisfied by cancellative semigroups and their groups of fractions,* (Preprint).

[8] A.I. Mal'cev, *Nilpotent Groups*, Ivanov Gos.Ped.Ins.Uc.zap (1953).

[9] W. McCune, Prover9, https://www.cs.unm.edu/~mccune/mace4/.

[10] G. I. Moghaddam and R. Padmanabhan, *Commutativity theorems for cancellative semigroups,* Semigroup Forum 95 (2017), no. 3, 448–454.

[11] G. I. Moghaddam, R. Padmanabhan, and Yang Zhang, *Automated reasoning with power maps.* Journal of Automated Reasoning, 64(4)(2020), 689–697.

[12] B.H. Neumann and T. Taylor, *Subsemigroups of Nilpotent Groups,* Proc. Roy. Soc. Ser. A274(1963), pp 1-4.

[13] T. Nordahi, *Semigroups satisfying* $(xy)^m = x^m y^m$, Semigroup Forum 8(4) (1974), 332–346.

[14] O. Ore, Linear Equations in Non-Commutative Fields Annals of Mathematics, Jul., 1931, Second Series, Vol. 32, pp.463-477.

[15] R. Padmanabhan, and Yang Zhang, *C*ommutativity theorems in groups with power-like maps, J. Formaliz. Reason. 12(1) (2019), 1–10.

[16] Chen-Te Yen, *Note on the commutativity of cancellative semigroups.* Bulletin of the Institute of Mathematics Academia Sinica, 10(2)(1982), 149-153.

[17] Chen-Te Yen, *On the commutativity of rings and cancellative semigroups.* Chinese Journal of Mathematics, 11(2)(1983), 99–113.

Software for indefinite integration

Corrundum.red – Integration for all

A. C. Norman[1], D. J. Jeffrey[2]

[1]Trinity College, Cambridge, U.K.

[2]Dept. Mathematics, The University of Western Ontario, London, Ontario, Canada

Abstract

The Rubi system for the evaluation of indefinite integrals was developed by Albert Rich over a period of more than 15 years. His unexpected death has led to a number of discussions on what can be done to develop the system further. One possible direction addresses the fact that Albert Rich's development was built entirely on the Mathematica system. A number of people have considered the porting of Rubi to a variety of alternative computer algebra systems. The requirements for porting are discussed.

Keywords

Indefinite integration, Anti-derivatives, Rule-based Integration, RUBI database, Reduce, Mathematica

1. Introduction

For over 80 years, a key resource regarding indefinite integration has been Gradshteyn and Ryzhik[1]. It has also been maintained and updated over the years, the 8th edition [2] being released in 2015. The modern alternative is one of the computer algebra systems. These have made major progress since the 1960s, when Slagle's SAINT[3] and Moses's SIN [4] were released. The web site 12000.org[5] reports testing 9 currently available systems[1] (commercial and public domain) against a test suite of 106812 integrals. As with any test suite, it can be criticized for showing biases, but it covers all the basic integrals and is of a scale that makes it hard to ignore. To a good approximation it is a concatenation of all the other significant sets of test cases that its authors could find.

The commercial systems have the highest success rates, with the open source systems significantly lower, save that the permissively licensed Rubi shows outstanding capability. We have therefore been studying Rubi to investigate the feasibility of its use with systems other than Mathematica, which is currently the foundation on which it is built.

We view this as being potentially valuable for four main reasons:

1. Many systems are still unable to find integrals for a significant fraction of the examples in the above torture test. This is likely to apply to almost every case that requires advanced special functions to express the result. Perhaps for many users this will not be a serious limitation, because their problem will not involve say the Fresnel S function. But measured against Gradshtein and Ryzhik it is a limitation.
2. Where results are generated they may often be bulkier than would be ideal. This can go as far as returning a result expressed using elliptic integrals or incorporating imaginary numbers when something more elementary would serve better.
3. Something that may count as a special case of the above is the treatment of multi-valued functions and delivering an integral in a form that handles them and their principal values consistently. This can be of extreme important in integration when users substitute in endpoints to find a definite integral.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
acn1@cam.ac.uk (A. C. Norman); djeffrey@uwo.ca (D. J. Jeffrey)
0000-0002-2161-6803 (D. J. Jeffrey)

[1]Mathematica, Maple, Mupad, Maxima, FriCas,Giac/Xcas, SymPy, Reduce, Rubi.

4. With reasonable generality, the existing schemes return integrals with no commentary available as to how they were derived, and the techniques used are embedded in large bodies of code which will be unavailable when one of the commercial systems is used, but opaque even if an open source system is selected. With Rubi a commentary on how results were obtained is almost automatically available.

2. What is Rubi?

The RUle-Based-Integration system Rubi is a public domain project addressing indefinite integration [6]. It consists of two parts. The first part is a database consisting of transformation rules which convert an integral expression into a simpler form, allowing an iterative application of the rules to evaluate the initially given integral. The second part is a collection of utility functions, at present coded in the Mathematica language, which massages mathematical expressions into forms suitable for the application of one of the transformation rules contained in the database.

Readers not familiar with Rubi may appreciate a little background concerning the project. Rubi was started by Albert Rich (1949–2023) circa 2007, after the development of the Derive computer algebra system was discontinued [7]. It is based on a term-rewriting rule-based paradigm, which was expected to bring the following benefits.

- Speed and compactness.
- An ability to display the steps taken during an evaluation.
- Results which are correct in the complex plane.
- Results which are in the most compact and æsthetically pleasing form possible.
- A system which is readily modified and extended.

Although Albert Rich corresponded with many others, and received their suggestions for integration rules, he largely worked alone, and as a consequence, only lightly documented his program code.

2.1. The current state of Rubi

There are presently two versions of Rubi available. The first is 4.16.1; this is the last version whose release was overseen by Albert Rich [6]. The other is 4.17.3, which has been retrieved from material left by Albert Rich and implemented by supporters of Rubi [8]. The versions are very similar and can be summarized as follows.

- There is a database containing over 7000 rules, recorded in Mathematica syntax. It is in the public domain. They rules can be read in mathematical notation in PDFs from the Rubi website [6].
- Rubi includes several test suites of integration problems. These form part of the test suite used by 12000.org.
- The order in which the rules are placed can have an effect on the performance.
- The system consists of more than the database of rules. There are large support files, also written in Mathematica syntax, defining many auxiliary functions. These functions serve a variety of roles.
  - Testing the properties of a parameter. For most integration problems, the parameters in the rules are given numerical values. Thus, it is assumed that a parameter can be tested to see whether it is positive, or an integer, and so on.
  - Applying an algebraic transformation, such as extracting the content of a polynomial, making a partial fraction expansion, etc.

  Unfortunately, there are very few comment lines in the files containing the auxiliary functions. This makes working with them difficult.

2.2. The way forward

It is unlikely that a person or persons will come forward and take over the development of Rubi. The attitude of the *de facto* custodians of Albert Rich's legacy is that Rubi is frozen in its current state. A person wishing to continue Rubi is allowed to use the database as they see fit, but any development will be a new project. The analogy being made is that anyone can read a scientific paper and then write their own paper with whatever new thoughts the author has. The author references and acknowledges the earlier work, but is *au fond* working on a separate project.

There are several ways in which Rubi can be developed further.

- More integration rules can be developed.
- There are a number of integrals in the test files which cause Rubi to disappear into a loop. A modification of the data base could reduce the occurrence of loops.
- Rubi could be transported to other languages, such as Maple, muPad, etc.

For those of us who do not have Mathematica installed on our computer (perhaps because of cost), or who really value being able to inspect the workings of software whose results we rely on, or who are developers of other algebra systems (present and future), Rubi looks a very attractive starting point. As already mentioned, at present Rubi 4.x.x exists in the form of a large set of Mathematica rewrite rules and a significant but not-too-large collection of utility code to back them up. Albert Rich, however, had been planning a transition to Rubi 5 that would transform the rule-set into a decision tree that would not rely to the same extent on pattern matching in the system it was installed on. That was expected to make it much easier to migrate Rubi for use with other systems. Unfortunately this part of the project remains incomplete.

The work explained here is called "Corrundum.red" and is a step towards making full Rubi available outside Mathematica. While it is being developed using the Reduce system[9][10], the intent is to keep reliance on the algebraic facilities in Reduce fairly low and reasonable well explained, so that adaptation elsewhere might be more readily possible. In particular the rule-rewrite engine does not rely on any such engine already built into Reduce; thus exactly what it does can be made explicitly visible.

The name "corrundum.red" is of course something of a pun, in that corrundum is crystalline aluminium oxide with hardness 9 on the Mohs scale, and when tainted with chromium so that it shows up as a vivid red colour you obtain the gemstone ruby. Of course ".red" is the file suffix used by the Reduce algebra system. So this name references Rubi (which is a sometimes-used alternate spelling for ruby); it links to Reduce and is intended to suggest both toughness and high value.

There have been multiple exercises to leverage the Rubi rule set in the past, so we are not the first to look into this. In fact, a previously abandoned investigation, using Reduce, was conducted by one of us (ACN).

We of course note the existence of "symja" or $S\Psi MJ\Lambda$ which aims mainly at support for Android and is all coded in Java. That has at its core an implementation of a language rather closely modelled on the one used by Mathematica, and as a result they have been able to import Rubi wholesale leading to very impressive capabilities. For our purposes this provides excellent confirmation that emulating enough of Mathematica to make Rubi work is feasible, and it provides a non closed-source version. But the Mathematica-style support there pervades so much of Symja that it does not help to guide those who want only enough of it for use in some alternative system that does not start off in Java and does not start off with Mathematica-like syntax and semantics baked in.

Especially with Albert Rich's work migrating Rubi from a rule-based system to one making transitions based on a decision tree, significant parts of Rubi were made to work on Maple, however again that project seems never to have reached a stage of full support for all the rules. https://de.maplesoft.com/company/casestudies/stories/maple-for-accuracy-and-support.aspx?L=G

Similarly Julia has the adoption of Rubi on its 2021 roadmap: https://juliasymbolics.org/roadmap/ but the github repository that we found https://github.com/ufechner7/Rubi.jl has been inactive for some years.

With all this background our work has many precedents. We hope that this paper will explain, in more detail than we were able to find elsewhere, some of the particular challenges in this activity, while we attempt to extricate Rubi from Mathematica syntax, semantics and support, and thus be of interest to others who want to revive previous attempts or start new ones.

Corrundum.red has a number of components and these will be explained in the following sections.

3. Capturing the Rubi rules

A parser for the subset of Mathematica syntax used in the transformation rules was coded for Reduce. Happily a rather naive recursive descent parser proved sufficient. Anyone at all familiar with this style of parser would observe that it is basically merely one procedure to correspond to each entry in the BNF rendition of the syntax involved. This naive parser was first created a decade ago and used to start an investigation of a version of Rubi from back then: that project stalled and the revived parser now had to have some extensions to cope with the Mathematica syntax that has meanwhile crept into Rubi and its utilities. Over that decade Reduce has been provided with a parser generator using the standard LALR scheme which could have made this scheme for reading Mathematica style text easier and much more compact, but re-using the existing almost-working if cruder scheme seemed the simpler path. And of course since this is just parsing expressions and not a full language the naive approach is not seriously offensive. We believe it would also have been possible to make Mathematica export files in a very similar format.

The result of a parse is the full set of rules expressed as Lisp-style data structures with a collection of operators that denote the various things that Mathematica provides. So for instance the very first Rubi rewrite starts as

```
(* ::Code:: *)
Int[u_.*(a_+b_.*x_^n_.)^p_.,x_Symbol] :=
  Int[u*(b*x^n)^p,x] /;
FreeQ[{a,b,n,p},x] && EqQ[a,0]
```

and becomes

```
(/;
  (:=
    (Int
      (times
        (_. u)
        (expt (plus (_ a) (times (_. b) (expt (_ x) (_. n)))) (_. p)))
      (_ x Symbol))
    (Int (times u (expt (times b (expt x n)) p)) x))
  (and (FreeQ (bracelist a b n p) x) (EqQ a 0)))
```

Actually in the file as generated all punctuation characters in the parenthesised form are preceded by escape characters so that for instance "/;" and "_." are seen as merely simple symbol names.

This adjusted format explicitly represents a parse tree and it is likely that internally Mathematica turns the raw input into something equivalent – but what we have here is totally unambiguous and entirely suitable for processing with any scheme that is happy to work with tree-like data structures. At the data structure level Reduce works with Lisp-style data and so it is especially comfortable with this, but it would be close to trivial to write code to read this tree notation in almost any language. Given that a simple tree-walk could re-display the material in the syntax most suitable for some other system, be it re-creating Mathematica syntax or converting for Maple, Maxima, Axiom or whatever.

It can be seen that a rule generally has three components. A template or pattern which here is basically $\int(u * (a + bx^n)^p, x)$ and where the various annotations are to mark various things as parameters or wildcards. Then there is the transformed version, which in this case purely serves to remove the a that is originally present. Finally there is a set of conditions. These indicate that the various parameters must all be independent of x and that this transformation that gets rid of a is only applicable when $a = 0$, in this case reminding us that the actual expression may contain a sub-expression equivalent to 0 but starting off looking more complicated. Of course we all know that identifying when a general expression is zero is undecidable!

There are almost 7500 of these rules! Using them involves facing up to (at least) two significant challenges. One is that of gaining a sufficiently full understanding of Mathematica pattern matching so that its behaviour against this rule-set can be re-created. The other is that throughout the Rubi rules there are function calls in both conditions and replacements that invoke a range of Mathematica primitives. Some (such as FreeQ here) have behavior that is simple to understand, but the precise expectations and limitations of even something as obvious looking as `EqQ[a,0]` have to cause some pause for thought.

4. Pattern matching and the Utilities

When one first admires Rubi, what most catches the eye is the set of simple-looking tree rewrite rules. These basically look at the structural form of integrands and on that basis perform transformations. Furthermore it is on record that Albert Rich's plans (for a Rubi 5) involved transforming the identification of which rules to apply into a decision tree – in effect a giant nest of "if" statements.

Thus when we started work, in a spirit of optimism we expected that, with the rule-set expressed as data structured such that Reduce could manipulate it easily, our main task would be to get simple tree-rewrites involving 7000+ rules to run reasonably efficiently.

After some while it became clear that things were not quite that straightforward.

1. When one writes `Int[expression, x]` in Mathematica, the expression delivered by Mathematica to pattern matching definitions of `Int` is neither the unaltered input nor a fully simplified version of it. It is somewhere in between. If Rubi is ported to any other world, this pre-processing is not likely to be replicated precisely, and so behaviour will differ. As a tiny example consider the Mathematica definition `f[a_*b_] := {"product", a, b}` which decomposes a product. The input `f[x+x]` reveals that what Mathematica sees for matching is the product of 2 and `x`, while `f[x*x]` is not seen as a product at all but as a power. `f[(1+x)*(x+1)]` is again seen as a power, while the more elaborate `f[x*((x+1)^2-1-x^2)]` is not seen as having an argument that is equivalent to x^2 and so is reported as being product. All of this is perfectly proper to the extent that it is Mathematica doing what it chooses to do, but it does make it harder to think about testing Rubi, since the exact form of expressions it sees have been subject to this not very clearly documented adjustment.
2. Mathematica pattern matching is powerful and it is aware that addition and multiplication are both commutative. Thus a pattern of the form `Int[a*x^2 + b*x + c, x]` (here the underscores are omitted to make the presentation neater) will match a quadratic regardless of the order in which the terms are presented. So in some sense a rule with that pattern as its left-hand side represents 6 cases corresponding to different term orderings. At one stage we considered pre-conditioning the rule set so that all of those cases were explicitly shown and hence the actual matching would not need to worry about this issue and hence might be simpler and faster: it became apparent that doing so caused the size of the rule set to explode beyond reason.
3. A further cleverness of the Mathematica matcher is that in certain cases some parameters can have default values. This is indicated by using `"_."` where they are mentioned in a pattern. With this scheme the pattern `a_. + b_.*x_^n_.` will match not just `p+q*z^k` but also variants on that with `p=0` and/or `q` and `k=1`, all the way down to just `z`. If one combines the consequence of this with commutativity this one pattern can match a dozen different input forms including say `z*q+p`. One rewrite rule can contain multiple instances of these two shorthands and we found that if everything was expanded out to show the full range of cases to be accepted that we could

end up with well over a thousand cases following from what was presented as a single rule. This astonished us!

4. The matching that is made has to be subject to some conditions - which are given at the end of the rule following the symbol `/;`. A first thought is to perform structural matching first and after that check if the constraints apply. However consider a case where the pattern is (without the underscores here) `a + b*x^n + c*x^m` and the constraints include `m=2*n`. Now commutativity shows that the two sub-patterns `b*x^n` and `c*x^m` could match input in either order, but only one is will suit the constraint. The most naive approach would be to insist that the syntactic matcher returned not just a single match but all possible ones that arise based on commutativity and default values and that the constraints are then used to determine which (if any) are viable. In some cases this would return very many matches. It feels more reasonable to make the structural matcher able to deliver its first match and then backtrack to look for another if that becomes necessary. Of course in the worst case just as many variants could have to be tried.

There are ways to address all of these issues. The pre-conditioning‘of input for instance to sort constant terms in sums and products either to the start or the end helps substantially with commutativity. Where we have arrived at is that we index constraints by the parameters they depend on. Then when our pattern matcher is about to ascribe a value to a parameter it checks if that completes the information about everything that the relevant constraints will use. It can thus test the conditions during matching and as soon as possible, and this of course saves us from investigating parts of an expression once we have discovered that that will be futile. With this we then accept the first viable ordering for any commuting argument list.

We are aware that our early-check scheme could be unsatisfactory for a fully general set of rules. Consider a complete pattern that contains both `(x^n+x^m)` and `(x^p+x^q)` and then pathologically a constraint on `n`, `m`, `p` and `q` that is not symmetric in them. The orderings for matching the two commuting sums would need to coordinate. We believe there are no such situations in the Rubi rule-set, so our matcher is not fully general but should cope with requirements here. This illustrates that an implementation of Mathematica-like matching that just has to cope with a fixed set of rules such as those in Rubi may be able to be simpler and hence potentially faster than one that needs to come with the most general case.

The matcher we have coded is essentially a modest size body of Lisp code and of itself it is independent of Reduce. It is intended to be reasonably clear and as concise as we can manage, so for instance to cope with a commutative operator it forms a list of all permutations and then scans the list reporting the first occasion it finds a match. It would clearly be possible to fold the enumeration of permutations with checking them but for a first version and for “work in progress” we preferred simplicity over potential performance gains. In a similar way and although we have thought of many ways that will allow us to do better, we have started with checking input expressions against each of the 7000+ patterns in turn and we have avoided exploring the interesting rabbit-hole of indexing, expression signatures and hashing, “trie” structures and expansion of the matching process into executable code. Those are all for further investigation at a later stage.

5. Predicate and Replacement material

On starting this project we expected that pattern matching would be the core of the work. Well, the conditions applied to rules contain many predicates that are obvious and easy to support. For instance for a very large proportion of parameters there is a check that the parameter does not depend on the independent variable. For exponents there can be checks such as that n is not -1. The test $m = 2 * n$ is slightly less obvious but once encountered is easy to handle. So at first we felt we were making good progress as we provided more and more of those. Within the formulae that appear as results there are cases where n is numeric and $n + 1$ is used, and it makes obvious sense to perform the arithmetic. Again that was easy. Perhaps especially so as it plays to Lisp’s strengths and is all code that could reasonably have appeared in 1960s attempts at integration! Following this path and by providing a tiny set of our own integration rules we were able to perform some first integrations really rather early in our work. At that time we felt greatly encouraged.

Beyond the trivial there are then a collection of Mathematica operations that do "genuine" algebra, but where the intent is clear and where any other host algebra system will have something equivalent. An example arises when a result will be of the form `Log[expression]` because if the expression has a constant factor (i.e. a non-trivial content with respect to the variable of integration) then that can be taken out and its only contribution to the result would be to merge it in with the constant of integration. So when Rubi uses a function `RemoveContent[]` that is liable to involve chaining to the underpinning algebra system and the exact details of how that is done can not be portable but the expected returned value is unambiguous.

Some Mathematica primitives as called here can of course represent special portability challenges and in the end the balance as to how much of Rubi's success flows from its own rule-set and how much from the power of (say) the Mathematica `Simplify[]` function (which will apply multiple transformations on its input and return the result that Mathematica things is nicest) is something that can not be answered until our work is complete.

However we then found that in both predicates and especially in replacements there were instances of function calls with significant depth. These are defined in a file `IntegrationUtilityFunctions.m`. The around 8000 lines of Mathematica code there include a great many small wrappers for simple operations that are not problematic at all, however there are other sections that gave us pause. An example is the function `Subst` which is fairly cryptically explained there as "Subst[u,x,v] returns u with all nondummy occurrences of x replaced by v and resulting constant terms replaced by 0." but where inspection reveals that it together with its immediate sub-functions represents around 800 lines of code. For a while that felt like a really severe road-block.

Our eventual response has been to recognize that it is necessary to view the Utilities file as a further set of rewrites to be handled alongside and using basically the same mechanisms as the main set of integration rules. This is possible because much of Mathematica's notation is close to being functional – all the "procedure definitions" in the Utilities file can, from only a slightly different perspective, be seen as "rewrites".

The Utilities use a distinctly broader range of Mathematica syntax and facilities than the main set of integration rules. This includes the various scoping constructs that interact with just how Mathematica interprets the meaning of symbols, catch and throw, mapping over lists with what are in effect lambda expressions and explicit in-line invocations of the pattern matcher. We have taken a pragmatic approach – when some utility function uses "fancy" Mathematica features but is in fact fairly small or simple we have just implemented a replacement in Lisp/Reduce so only large and messy functions need to be interpreted from their Mathematica form. We end up with what amounts to an interpreter for just enough of the Mathematica language to cope with Rubi, and a scheme where any particular function might end up either recursing in the interpreter to continue processing Mathematica forms or dropping out into our own independent code.

As we were working on that we had arranged that any attempt to use a function that had not yet been provided would lead to a clear message but that processing could continue (with Rubi typically just considering the rewrite concerned inapplicable and going on to try the next). This meant that by running the full almost 80K test cases through our code we could count which functions were blocking progress most often. At all stages this provided us with a form of priority list. For finer grain investigation we obviously also maintained scripts to test just a sample from the full test suite and also to try our own hand-picked cases so that they could be run with extra tracing.

For longer than we had imagined would be the case our corrundum.red could only deliver results for around 1% of the examples in the test suite and we perhaps started to feel a little despondent. On implementing some more Mathematica functions to get the Utilities going at one stage boosted that so we could solve almost 7% of the first 10000 test cases. The current state is that our code now does not report any dangling unimplemented functions but clearly the emulation of some of Mathematica is still buggy and for most examples the code yields a result that is a malformed formula. This is liable to be "just a bug" and so we hope it will soon be fixed! But this paper is explicitly about "work in progress" and the comments here show that that is exactly where we are.

6. Current Results

On testing cases from the Rubi test suite our performance so far is quite lamentable, with the very best try to date managing to get proper answers for just under 7% of the cases! But this mainly reflects some residual Mathematica primitives that have not yet been coded and debugged enough to let all examples run to completion. However our code can show the transformations that it makes and so one can observe the rewrite engine both when behaving well and when not.

Here is a case of mixed gladness, where there is some degree of mixture between Mathamatica botation (eg Log) and Reduce (eg log):

```
Test case 57
Use rule 2778 on Int(Log(x),x)
transforms to x*Log(x) + (-1)*x
Use rule 2779 on Int(Log(x)^2,x)
transforms to x*Log(x)^2 + (-1)*xtimes(2,x*Log(x) + (-1)*x)
(~~~ test case 57 My leafcount 19 Theirs 15)
Input:

int(log(x)^2,x)
My result:

x*log(x)^2 + (-1)*xtimes(2,x*log(x) + (-1)*x)
Reference result from Rubi test file:

2*x + (-2)*x*log(x) + x*log(x)^2
```

"xtimes" is our rendering of the symbol \[Star] that Rubi introduces in some places where we suspect it is trying to prevent Mathematica from transforming products under its feet. You can see that at present we have not turned it back into ordinary multiplication on the way out. But you can also see that along the way Corrundum has used two of the Rubi rules (which we call numbers 2778 and 2779, those numbers being easily decoded in our source files) and that despite its ugliness our result here is not much bulkier than the "official" Rubi one and is basically correct. This and many other cases illustrates that the exact form of the official Rubi result depends on what amounts to post-Rubi term reordering and cleanup done by Mathematica, because in those cases our results can be equally correct but just (for instance) order terms differently.

Various of the more severe problem cases we face are going to be easy to track and trace from the sort of output we generate: first we show the input expression, then for each pattern that matches we document both the identity of the rule triggered and the resulting transformed expression. Given that this is all basically Lisp code it is then simple to set tracing on the functions that implement each step.

The above statement is correct in that debugging steps that are taken can be straightforward – however there may also be instances in which Corrundum fails to make a match that Rubi expects and perhaps relies on. This could either be because of limitations in our pattern matching code or because intermediate transforms have not left an expression in exactly the shape that Rubi expected - for instance because reaching that shape was a consequence of fine details of Mathematica processing.

To help us track issues of that sort we are taking a second track. We have taken the full set of Rubi rules in Mathematica format and added annotations that cause Mathematica to report on each rewrite it makes, as in:

```
Test case 2: Int[x Sqrt[1 + 3 x], x]
```

```
Rule 120
Rule 125
...
Rule 105
Rule 104

                     3/2                 5/2
         -2 (1 + 3 x)       2 (1 + 3 x)
Result: -------------- + -------------
                27                 45
```

and by cross referencing this against output from our system we will be able to track rewrites that we do not make as well as debug ones that we do. Again the numbers attached to rules are ones that have to be interpreted by indexing into our versions of the source files – they are not absolutely fixed to the reference version of the Rubi sources. Also the rule-tracing is not the one that Rubi has as one of its nicer features - it is one that provides yet finer grain information about the rewrites that are applied.

This sort of trace also makes it possible to identify the modest number of cases where the current Rubi rule-set can lead to unending cycles of transformation, so that attention can be given to tidying up there. It also has the potential to let us identify both which Rubi rules are used most (so we can improve performance by checking for those ones first) and perhaps spotting ones that are never used at all.

As has been notes several times already this is a project still in full swing, but we believe it has already hit a number of valuable goals:

1. In understanding the current state of Rubi;
2. Understanding much more about the way in which it intertwines with Mathematica than any experiment that remained within the Mathematica world could;
3. Provide a somewhat freestanding rewrite engine to use the Rubi rules. The dependence we have on Reduce for algebra support is fairly modest (at least in our opinion) and (maybe when we get round to it) easy to document. The main engine we have is expressed as Lisp code (albeit presented in Reduce syntax that sugars it and may make it easier for outsiders to come to grips with it;
4. Get a system that can perform some integration using the Rubi rule-set, with a real prospect that getting a lot further is now "just debugging". We have a "proof of concept".
5. As we have developed this we have stashed away multiple ideas for performance engineering to make our system rather fast.

So this has been and remains an entertaining project to have been working on, and we have now reached a state where we would be willing to get others with suitable coding competence to join in to push the work further, either until we have something fit to merge into Reduce or to build on what we have done to consider support for say Maxima, Maple or some other system.

7. Some lessons mainly about Mathematica that we have learned

The characterisation here comes from the perspective of one who is unfamiliar with Mathematica and only confronts it through the prism of Rubi. This will of course provide a tinted and distorted impression, but it is nevertheless relevant while trying to reproduce Rubi's behaviour. So here we collect points, some of which have been alluded to earlier but are still things we know now that we did not at the start of this work.

To a first approximation Mathematica has two schemes for processing algebraic expressions. There seems to be a range of transformations that are automatically and universally made. These include cases like $1+1 \rightarrow 2$ but also $a*z*a \rightarrow a^2*z$ where terms in products are sorted and consolidated into powers when they are sufficiently identical, but not if they are equivalent but sufficiently textually different. There are then a wide range of functions that can explicitly perform operations such as expanding products, arranging a common denominator and so on. Finally there are generic simplification functions that try out a larger or smaller number of the explicit transformations and return what they judge to be the nicest variant on their input that they come across.

We explain this here because a study of the Rubi sources shows clearly that Albert Rich found the need to work around some of the details of all this. Two cases in particular illustrate this by showing transformations only reasonably comprehensible as work-arounds to detailed Mathematica behaviour,

In Mathematica the standard trigonometric functions are named Sin, Cos, Tan and the like - with upper case initial letters. The Rubi rules and utilities in places introduce a parallel set called sin, cos and tan in lower case that are referred to as "inert". Rules first convert from the standard Mathematica versions to these ones and apply their own set of simplifications, but also at least potentially pass the inert expressions through more general Mathematica transformations where presumably otherwise the system would have "simplified" things in unwanted ways.

Somewhat similarly \[Star] is used for a variant on multiplication which is processed a little differently from the regular Mathematica one.

Finally the utility functions file contains a definition

```
FixIntRules[rulelist_] := Block[{Int, Subst, Simp, Star},
  SetAttributes[{Int, Subst, Simp, Star},HoldAll];
  Map[Function[FixIntRule[#,#[[1,1,2,1]]]], rulelist]]
```

where `SetAttributes` seems to be being used to make fairly broad adjustments to Mathematica processing in a way that anybody not familiar with both all the Rubi rules and their intended interactions and just what Mathematica will do with or without that directive may find challenging to decode.

A feeling that arises from all of this is that to support Rubi *fully* outside Mathematica it is perhaps necessary to emulate some of the corners of Mathematica behaviour where Rubi is then carefully taking steps to allow for the fact that it did not want them to apply!

None of this is to be viewed as a criticism of either Mathematica or of that state of the final Rubi snapshot, but it may suggest that the entanglement between them is non-trivial. A different view is that perhaps in reality the worries expressed here will end up impacting only a small fraction of the examples in the test suite, and in some future Rubi some adaptation or adjustment of rules may sort that out easily. Until our work is complete we can not tell!

References

[1] I. S. Gradshteyn, I. M. Ryzhik, Table of Integrals, Series, and Products, Gosudarstvennoe Izdatel'stvo Tehniko-Teoretieskoj Literatury, 1943.

[2] I. S. Gradshteyn, I. M. Ryzhik, D. Zwillinger, V. Moll, Table of integrals, series, and products; 8th ed., Academic Press, Amsterdam, 2015. URL: https://cds.cern.ch/record/1702455. doi:0123849330.

[3] J. Slagle, A heuristic program that solves symbolic integration problems in freshman calculus : symbolic automatic integrator (SAINT), Ph.D. thesis, MIT, 1961.

[4] J. Moses, Symbolic Integration, Technical Report AC-TR-47, MIT, 1967.

[5] N. M. Abbas, Cas integration tests, 2024. https://www.12000.org/my_notes/CAS_integration_tests/index.htm.

[6] A. D. Rich, P. Scheibe, Rubi (Rule-based Integrator), 2024. https://rulebasedintegration.org/.

[7] A. D. Rich, D. J. Jeffrey, A knowledge repository for indefinite integration based on transformation rules, in: Intelligent Computer Mathematics – LNCS 5625, Springer, 2009, pp. 480–485.

[8] A. Rich, P. Scheibe, Rubi – rule-based integration, 2024. https://github.com/RuleBasedIntegration/Rubi/releases/tag/4.17.3.0.

[9] A. C. Hearn, REDUCE: A user-oriented interactive system for algebraic simplification, in: M. Klerer, J. Reinfelds (Eds.), Interactive Systems for Experimental Applied Mathematics, Academic Press, New York, 1968, pp. 79–90.

[10] A. C. Hearn, many contributors, Reduce – a portable general-purpose computer algebra system, 2024. https://sourceforge.net/projects/reduce-algebra.

Towards Trajectory Planning for a 6-Degree-of-Freedom Robot Manipulator Considering the Orientation of the End-effector Using Computer Algebra*

Takumu Okazaki[1], Akira Terui[2,*] and Masahiko Mikawa[3]

[1]Master's Program in Mathematics, Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8571, Japan

[2]Institute of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan

[3]Institute of Library, Information and Media Science, University of Tsukuba, Tsukuba 305-8550, Japan

Abstract

We consider the trajectory planning of a 6-Degree-of-Freedom (DOF) robot manipulator using computer algebra, with controlling the orientation of the end-effector. As a first step towards the objective, we present a solution to the inverse kinematics problem of the manipulator such that the orientation of the end-effector remains constant using computer algebra.

Keywords

Trajectory planning, Inverse kinematic problem, Robot manipulator, Gröbner basis

1. Introduction

This paper discusses the trajectory planning of a 6-Degree-of-Freedom (DOF) robot manipulator. A manipulator is a robot resembling a human hand and comprises links that function as a human arm and joints that function as human joints. Each link is connected for movement relative to each other by a joint. The first link is connected to the ground, and the last link called end-effector, contains the hand, which can be moved freely. In this paper, we consider a manipulator called "myCobot 280" [1] (hereafter called "myCobot") that has six joints connected in series that can only rotate around a certain axis. Note that each joint has one degree of freedom. Therefore, myCobot has at most six degrees of freedom.

The inverse kinematics problem is a problem of determining the joint arrangement when the end-effector is placed in a specified direction on a certain coordinate in space. The trajectory planning problem of the manipulator is an inverse kinematics problem in which the position of the end-effector is expanded from a single coordinate to a trajectory. In other words, it can be regarded as a problem to find the displacement of the joint when the end-effector of the manipulator moves on a given trajectory from the initial position to the final position.

In computer algebra, methods for the inverse kinematics problem of a 6-DOF robot manipulator have been proposed for more than 30 years ([2], [3]). Furthermore, several methods have been proposed to solve the inverse kinematics problem of a manipulator using Gröbner basis computation ([4], [5], [6], [7]). Among them, two of the present authors have proposed methods for solving the inverse kinematic problem ([8], [9]) and trajectory planning problem ([10]) of a 3-DOF manipulator using computer algebra. In more detail, we have proposed a method for solving the inverse kinematic problem efficiently with the use of Comprehensive Gröbner

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

*This work was partially supported by JKA and its promotion funds from KEIRIN RACE.

*Corresponding author.

s2320132@u.tsukuba.ac.jp (T. Okazaki); terui@math.tsukuba.ac.jp (A. Terui); mikawa@slis.tsukuba.ac.jp (M. Mikawa)

https://researchmap.jp/aterui (A. Terui); https://mikawalab.org/ (M. Mikawa)

0000-0003-0846-3643 (A. Terui); 0000-0002-2193-3198 (M. Mikawa)

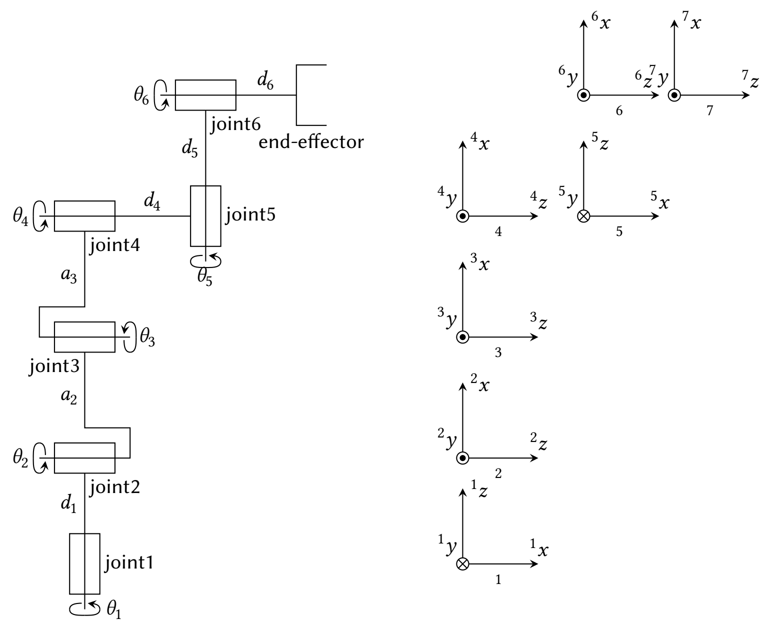


Figure 1: The schematic diagram and the local coordinate systems of each joint in myCobot.

Systems (CGS) and certifying the existence of a solution to the inverse kinematic problem using the CGS-QE, or the quantifier elimination (QE) based on the CGS computation.

In this paper, we consider the trajectory planning of a 6-DOF robot manipulator while controlling the orientation of the end-effector using computer algebra. As a first step towards the objective, we present a solution to the inverse kinematics problem such that the end-effector's orientation remains constant. More precisely, we present a solution to the inverse kinematics problem under the condition that the end-effector's local coordinate system overlaps the global coordinate system by translation.

The paper is organized as follows. In Section 2, the coordinate system and the notation of the manipulator are introduced. In Section 3, the analytical solution of general inverse kinematics problems is first described, followed by the solution in myCobot. In Section 4, future initiatives are described.

2. Preliminaries

2.1. Coordinate systems

For each joint in myCobot, the coordinate system is defined as follows (see Figure 1). Let each joint be numbered as joint i from the base to the end-effector in increasing order, and the end-effector be numbered as joint 7. Let $_i$ be the coordinate system of joint i. Here, $_1$ is the reference (global) coordinate system, while the other coordinate systems are local. The axes of each coordinate system are defined as follows: the ${}^iz$ axis in the direction of the joint axis (rotational axis in the case of a revolute joint), ${}^ix$ axis in the direction of the common normal of the ${}^iz$ and ${}^{i+1}z$ axes, and ${}^iy$ axis to be a right-handed system.

2.2. Notation

For the index i, the matrices are denoted by A_i, and vectors are denoted by ${}^{i+1}\underline{x}_n$, in which subscripts are used to distinguish points, and superscripts denote the local coordinate system to which they refer (vectors with no superscripts are referenced in $_1$). If the vector is referenced with respect to a different coordinate system, it is enclosed within brackets, and a separate superscript is added outside the brackets. (e.g. ${}^{i}[{}^{i+1}\underline{x}_n]$). Scalars are expressed in lowercase variables, and subscripts, such as a_i, are used where necessary. In $_i$, the origin is denoted by O_i and the unit vectors are denoted by ${}^{i}\underline{e}_x$, ${}^{i}\underline{e}_y$ and ${}^{i}\underline{e}_z$. In vector $\underline{v}$, the i-th component is denoted by $\underline{v}[i]$.

2.3. The Denavit-Hartenberg convention

The 'Denavit-Hartenberg parameters' [11] are used as a transformation method between coordinate systems. The following parameters are used in the transformations: a_i as the length of the common normal between the ${}^{i}z$ and ${}^{i+1}z$ axes, α_i as the angle from the ${}^{i}z$-axis towards the ${}^{i+1}z$-axis around the ${}^{i+1}x$-axis in the clockwise direction, d_i as the length between the common normal a_i and the origin of the coordinate system i, and θ_i as the angle between the common normal a_i and the ${}^{i}x$ axis.

Let A_i be the transformation matrix from $_{i+1}$ and $_i$. Then, as the product of matrices representing rotation and translation, A_i is expressed by using the above parameters as

$$A_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1}$$

For expressing the transformation of the coordinate system, affine coordinates are used as follows: if the coordinates of a point is expressed as ${}^{t}({}^{i}x, {}^{i}y, {}^{i}z)$ and also as ${}^{t+1}({}^{i+1}x, {}^{i+1}y, {}^{i+1}z)$, they are denoted by ${}^{i}\underline{X} = {}^{t}({}^{i}x, {}^{i}y, {}^{i}z, 1)$ and ${}^{i+1}\underline{X} = ({}^{i+1}x, {}^{i+1}y, {}^{i+1}z, 1)$, respectively, in which the last coordinates represent translation. Then, by the definition of A_i, we have ${}^{i}\underline{X} = A_i{}^{i+1}\underline{X}$ and ${}^{i+1}\underline{X} = A_i^{-1}{}^{i}\underline{X}$, where

$$A_i^{-1} = \begin{pmatrix} \cos\theta_i & \sin\theta_i & 0 & -a_i \\ -\sin\theta_i\cos\alpha_i & \cos\theta_i\cos\alpha_i & \sin\alpha_i & -d_i\sin\alpha_i \\ \sin\theta_i\sin\alpha_i & -\cos\theta_i\sin\alpha_i & \cos\alpha_i & -d_i\cos\alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

If $n + 1$ coordinate systems are given, there exist n coordinate transformations between neighboring coordinate systems. Therefore, if the coordinates of a point are given as ${}^{7}\underline{X}$ with respect to $_7$ (the end-effector), the coordinates ${}^{1}\underline{X}$ of the same point with respect to $_1$ (the global coordinate system) is obtained by multiplying A_i in sequence, such that

$${}^{1}\underline{X} = \mathrm{Aeq}\,{}^{7}\underline{X}, \quad \mathrm{Aeq} = A_1A_2A_3A_4A_5A_6. \tag{2}$$

Note that the inverse transformation is given as $\mathrm{Aeq}^{-1} = A_6^{-1}A_5^{-1}A_4^{-1}A_3^{-1}A_2^{-1}A_1^{-1}$.

3. Solving the inverse kinematic problem

3.1. The forward kinematics

We first consider the forward kinematics problem of myCobot. Let $\underline{p} = \overrightarrow{O_1O_7}$ be the vector from the origin of $_1$ (position of the root of the manipulator) to the origin of $_7$ (position of the end-effector), and let $\underline{l} = {}^{t}(l_1, l_2, l_3) = {}^{1}[{}^{7}\underline{e}_x]$, $\underline{m} = {}^{t}(m_1, m_2, m_3) = {}^{1}[{}^{7}\underline{e}_y]$, $\underline{n} = {}^{t}(n_1, n_2, n_3) = {}^{1}[{}^{7}\underline{e}_z]$ be the vectors that are parallel to the unit vectors on the x, y, and z axes of $_7$, respectively. Then, $\underline{p}, \underline{l}, \underline{m}, \underline{n}$ is represented with respect to $_1$ as

$$\begin{pmatrix} l_1 \\ l_2 \\ l_3 \\ 0 \end{pmatrix} = \mathrm{Aeq}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ 0 \end{pmatrix} = \mathrm{Aeq}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} n_1 \\ n_2 \\ n_3 \\ 0 \end{pmatrix} = \mathrm{Aeq}\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \mathrm{Aeq}\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

therefore, each component of Aeq and Aeq^{-1} is obtained as

$$\mathrm{Aeq} = \begin{pmatrix} l_1 & m_1 & n_1 & p_1 \\ l_2 & m_2 & n_2 & p_2 \\ l_3 & m_3 & n_3 & p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathrm{Aeq}^{-1} = \begin{pmatrix} l_1 & l_2 & l_3 & -(l_1p_1 + l_2p_2 + l_3p_3) \\ m_1 & m_2 & m_3 & -(m_1p_1 + m_2p_2 + m_3p_3) \\ n_1 & n_2 & n_3 & -(n_1p_1 + n_2p_2 + n_3p_3) \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3}$$

As seen in eq. (3), by putting the angles of the joints $\theta_1, \ldots, \theta_6$ into the transformation matrix A_i, the position and orientation of the end-effector are obtained.

3.2. The inverse kinematic problem

To make use of eq. (3) in myCobot, we substitute some of the joint parameters $a_i, \alpha_i, d_i, \theta_i$ of myCobot into the transformation matrix A_i. As shown in the schematic diagram of myCobot in Figure 1, where the squares represent the revolute joints and the angle of rotation θ_i is given by taking a positive counterclockwise direction with respect to the ${}^iz$ axis, the joint parameters are given as

$$\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\} = \{\pi/2, 0, 0, \pi/2, \pi/2, 0\}, \quad a_1 = d_2 = d_3 = a_4 = a_5 = a_6 = 0. \tag{4}$$

By substituting the joint parameters in eq. (4) into the transformation matrix A_i in eq. (1), Aeq is calculated. Then, by comparing the components of Aeq with the components of Aeq in eq. (3), a system of 12 polynomial equations in the variables s_i and c_i $(i = 1, \ldots, 6)$ is obtained, where $s_i = \sin\theta_i$ and $c_i = \cos\theta_i$. Each θ_i can be obtained by selecting the appropriate equation(s) from the above system and finding the solution. However, in the computation of Gröbner basis, the coefficients of the equations may expand, which makes it difficult to find the solution. Therefore, we focus on the structure of myCobot and solve its inverse kinematic problem from a different perspective as follows.

3.3. Solving the inverse kinematic problem with a fixed orientation

Inverse kinematics problems for robot manipulators of a certain structure can be solved analytically [12, 13]. Pieper [13] has pointed out that when the end effector of a 6-DOF manipulator has a spherical joint, the inverse kinematics problem can be separated into position and orientation problems of the end-effector. He has further noted that when the rotational axes of three consecutive rotational joints of a 6-DOF manipulator intersect at a single point, the inverse kinematics problem can also be separated into position and orientation problems of the end-effector.

Pieper's argument suggests that if there exists a combination of three consecutive rotational joints whose rotational axes intersect at a single point, it becomes possible to solve the inverse kinematics problem analytically. However, unfortunately, in the case of myCobot, there are no combinations of three consecutive rotational joints whose rotational axes intersect at a single point, although there are combinations of two consecutive joints whose rotational axes intersect at a single point. Therefore, following Pieper's approach, we solve the inverse kinematics problem by imposing constraints on the orientation of the end-effector.

For simplicity, we solve the inverse kinematic problem with the condition that the orientation of the end-effector remains constant, i.e. the axis in ${}_7$ is parallel to the axis in ${}_1$ preserving the same direction. Let $\underline{l} = {}^t(1, 0, 0), \underline{m} = {}^t(0, 1, 0), \underline{n} = {}^t(0, 0, 1)$, then Aeq and Aeq^{-1} in eq. (3) become as

$$\mathrm{Aeq} = \begin{pmatrix} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_2 \\ 0 & 0 & 1 & p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \mathrm{Aeq}^{-1} = \begin{pmatrix} 1 & 0 & 0 & -p_1 \\ 0 & 1 & 0 & -p_2 \\ 0 & 0 & 1 & -p_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Let P be the intersection point of axes ${}^4z$ and ${}^5z$, expressed as

$${}^1\underline{P} = \overrightarrow{O_1P} = {}^t(x, y, z, 1). \tag{5}$$

Note that P is the position of Joint 5. We first express $\sin\theta_i$ and $\cos\theta_i$ $\quad (i = 1, \dots, 6)$ with the coordinate of the intersection P.

Let ${}^7\underline{P} = \overrightarrow{O_7 P}$, then ${}^7\underline{P}$ is expressed in two ways as

$${}^7\underline{P} = A_6^{-1} A_5^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -d_5 \sin\theta_6 \\ -d_5 \cos\theta_6 \\ -d_6 \\ 1 \end{pmatrix}, \quad {}^7\underline{P} = \mathrm{Aeq}^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x - p_1 \\ y - p_2 \\ z - p_3 \\ 1 \end{pmatrix}.$$

By equating two vectors ${}^7\underline{P}$ above, we have

$$\sin\theta_6 = \frac{p_1 - x}{d_5}, \quad \cos\theta_6 = \frac{p_2 - y}{d_5}. \tag{6}$$

Let $\underline{w}_4$ be the unit vector in the direction of ${}^4z$-axis. Then, $\underline{w}_4$ and ${}^7\left[\underline{w}_4\right]$ are expressed as

$$\underline{w}_4 = A_1 A_2 A_3 \begin{pmatrix} \sin\theta_1 \\ -\cos\theta_1 \\ 0 \\ 0 \end{pmatrix}, \quad {}^7\left[\underline{w}_4\right] = A_6^{-1} A_5^{-1} A_4^{-1} \begin{pmatrix} \cos\theta_6 \sin\theta_5 \\ -\sin\theta_6 \sin\theta_5 \\ -\cos\theta_5 \\ 0 \end{pmatrix}. \tag{7}$$

By using ${}^7\left[\underline{w}_4\right]$ above and Aeq, we obtain another expression for $\underline{w}_4$ as

$$\underline{w}_4 = \mathrm{Aeq}^7 \left[\underline{w}_4\right] = {}^t(\cos\theta_6 \sin\theta_5, -\sin\theta_6 \sin\theta_5, -\cos\theta_5, 0). \tag{8}$$

By comparing each component of $\underline{w}_4$ in eqs. (7) and (8), respectively, we have

$$\cos\theta_5 = 0, \quad \sin\theta_5 = \pm 1, \quad \cos\theta_1 = \pm\sin\theta_6 = \pm\frac{p_1 - x}{d_5}, \quad \sin\theta_1 = \pm\cos\theta_6 = \pm\frac{p_2 - y}{d_5}. \tag{9}$$

Next, $\underline{P}_3 = \overrightarrow{O_1 P}$ is expressed as

$$\underline{P}_3 = A_1 A_2 A_3 \begin{pmatrix} 0 \\ 0 \\ d_4 \\ 1 \end{pmatrix} = \begin{pmatrix} d_4 \sin\theta_1 + a_2 \cos\theta_1 \cos\theta_2 + a_3 \cos\theta_1 (\cos\theta_2 \cos\theta_3 - \sin\theta_2 \sin\theta_3) \\ -d_4 \cos\theta_1 + a_2 \sin\theta_1 \cos\theta_2 + a_3 \sin\theta_1 (\cos\theta_2 \cos\theta_3 - \sin\theta_2 \sin\theta_3) \\ d_1 + a_2 \sin\theta_2 + a_3 (\sin\theta_2 \cos\theta_3 + \cos\theta_2 \sin\theta_3) \\ 1 \end{pmatrix}. \tag{10}$$

On the other hand, $\underline{P}_3$ is also expressed as ${}^1\underline{P} = {}^t(x, y, z, 1)$ as shown in eq. (5). Let $Q = {}^1\underline{P}[1]^2 + {}^1\underline{P}[2]^2 + ({}^1\underline{P}[3] - d_1)^2 = \underline{P}_3[1]^2 + \underline{P}_3[2]^2 + (\underline{P}_3[3] - d_1)^2$, then we have

$$Q = x^2 + y^2 + (z - d_1)^2 = a_2^2 + a_3^2 + d_4^2 + 2 a_2 a_3 \cos\theta_3. \tag{11}$$

Therefore,

$$\cos\theta_3 = \frac{x^2 + y^2 + (z - d_1)^2 - a_2^2 - a_3^2 - d_4^2}{2 a_2 a_3}, \quad \sin\theta_3 = \pm\sqrt{1 - (\cos\theta_3)^2}$$

From the third component of $\underline{P}_3$ and the constraints,we have

$$d_1 + a_2 \sin\theta_2 + a_3 (\sin\theta_2 \cos\theta_3 + \cos\theta_2 \sin\theta_3) = z, \tag{12}$$

$$(\sin\theta_2)^2 + (\cos\theta_2)^2 = 1. \tag{13}$$

If $\cos\theta_3, \sin\theta_3$ is obtained, $\cos\theta_2, \sin\theta_2$ is easily found from eq. (12).

Finally to find $\cos\theta_4$ and $\sin\theta_4$, let $\underline{w}_5$ be the unit vector in the direction of ${}^5z$-axis. Then, by using ${}^7\left[\underline{w}_5\right]$ expressed as

$${}^7[\underline{w}_5] = A_6^{-1} A_5^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \sin\theta_6 \\ \cos\theta_6 \\ 0 \\ 0 \end{pmatrix},$$

we have the expression for $\underline{w}_5$ in two ways as

$$\underline{w}_5 = A_1A_2A_3A_4\begin{pmatrix}0\\0\\1\\0\end{pmatrix} = \begin{pmatrix}\cos\theta_1\sin(\theta_2+\theta_3+\theta_4)\\ \sin\theta_1\sin(\theta_2+\theta_3+\theta_4)\\ -\cos(\theta_2+\theta_3+\theta_4)\\ 0\end{pmatrix}, \quad \underline{w}_5 = \mathrm{Aeq}^7[\underline{w}_5] = \begin{pmatrix}\sin\theta_6\\ \cos\theta_6\\ 0\\ 0\end{pmatrix}.$$

Then, by comparing each component of $\underline{w}_5$ above, θ_4 is obtained as the one satisfying $\cos(\theta_2 + \theta_3 + \theta_4) = 0$ and $\sin(\theta_2 + \theta_3 + \theta_4) = \pm 1$, together with the use of the additivity theorem.

We could represent $\sin\theta_i$ and $\cos\theta_i \quad (i = 1, \dots, 6)$ using the coordinates of the intersection P. Next, we want to find x, y and z.

First, comparing the third component of ${}^7\underline{P}$, we have

$$-d_6 = z - p_3. \tag{14}$$

Next, from eq. (6) and the trigonometric identity, we have

$$\left(\frac{p_1 - x}{d_5}\right)^2 + \left(\frac{p_2 - y}{d_5}\right)^2 = 1. \tag{15}$$

Finally, by equating the first and the third components in the vector in the right-most-hand of eq. (10) with x and z, respectively, we have

$$\begin{aligned} d_4\sin\theta_1 + a_2\cos\theta_1\cos\theta_2 + a_3\cos\theta_1(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) &= x,\\ d_1 + a_2\sin\theta_2 + a_3(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) &= z. \end{aligned}$$

Then, by substituting $\sin\theta_1 = \frac{P_2 - y}{d_5}$ and $\cos\theta_1 = \frac{p_1 - x}{d_5}$ in eq. (9) into the above equations, assuming first that $\sin\theta_5 = 1$, we have the following system of equations in $\sin\theta_2, \cos\theta_2, \sin\theta_3, \cos\theta_3$:

$$\begin{aligned} d_4\frac{p_2 - y}{d_5} + a_2\frac{p_1 - x}{d_5}\cos\theta_2 + a_3\frac{p_1 - x}{d_5}(\cos\theta_2\cos\theta_3 - \sin\theta_2\sin\theta_3) &= x,\\ d_1 + a_2\sin\theta_2 + a_3(\sin\theta_2\cos\theta_3 + \cos\theta_2\sin\theta_3) &= z,\\ (\sin\theta_2)^2 + (\cos\theta_2)^2 &= 1,\\ (\sin\theta_3)^2 + (\cos\theta_3)^2 &= 1, \end{aligned} \tag{16}$$

in which the last two equations are added as trigonometric identities.

Then, the solution of the system in eq. (16) gives the value of $\cos\theta_3$ as

$$\begin{aligned} \cos\theta_3 = \frac{1}{2a_2a_3(p_1 - x)^2}\big(&-a_2^2p_1^2 - a_3^2p_1^2 + p_1^2d_1^2 + p_2^2d_4^2 + 2a_2^2p_1x + 2a_3^2p_1x - 2p_1d_1^2x - 2p_2d_4d_5x\\ &- a_2^2x^2 - a_3^2x^2 + d_1^2x^2 + d_5^2x^2 - 2p_2d_4^2y + 2d_4d_5xy + d_4^2y^2 - 2p_1^2d_1z + 4p_1d_1xz - 2d_1x^2z\\ &+ p_1^2z^2 - 2p_1xz^2 + x^2z^2\big). \end{aligned} \tag{17}$$

By putting $\cos\theta_3$ in eq. (17) into the first equation in eq. (11) and multiplying both sides by $(p_1 - x)^2$, we have

$$\begin{aligned} &d_4^2p_1^2 + d_4^2p_2^2 - 2(d_4^2p_1 + d_4d_5p_2)x + (d_4^2 + d_5^2 - p_1^2)x^2 + 2p_1x^3 - x^4\\ &\qquad - 2d_4^2p_2y + 2d_4d_5xy + (d_4^2 - p_1^2)y^2 + 2p_1xy^2 - x^2y^2 = 0. \end{aligned} \tag{18}$$

In the case $\sin\theta_5 = -1$, perform the same calculation with $\sin\theta_1 = \frac{P_2 - y}{d_5}$ and $\cos\theta_1 = \frac{p_1 - x}{d_5}$. In this case, the sign in eqs. (17) and (18) changes in part, which gives the following equations.

$$\cos\theta_3 = \frac{1}{2a_2a_3(p_1-x)^2}\big(-a_2^2p_1^2 - a_3^2p_1^2 + p_1^2d_1^2 + p_2^2d_4^2 + 2a_2^2p_1x + 2a_3^2p_1x - 2p_1d_1^2x + 2p_2d_4d_5x$$
$$- a_2^2x^2 - a_3^2x^2 + d_1^2x^2 + d_5^2x^2 - 2p_2d_4^2y - 2d_4d_5xy + d_4^2y^2 - 2p_1^2d_1z + 4p_1d_1xz - 2d_1x^2z$$
$$+ p_1^2z^2 - 2p_1xz^2 + x^2z^2\big), \quad (19)$$

$$d_4^2p_1^2 + d_4^2p_2^2 - 2(d_4^2p_1 - d_4d_5p_2)x + (d_4^2 + d_5^2 - p_1^2)x^2 + 2p_1x^3 - x^4$$
$$- 2d_4^2p_2y - 2d_4d_5xy + (d_4^2 - p_1^2)y^2 + 2p_1xy^2 - x^2y^2 = 0. \quad (20)$$

Furthermore, eq. (18) or eq. (20) together with eqs. (14) and (15), we obtain a system of polynomial equations in x, y, z. Solving the system (by using Gröbner basis computation, etc.) gives the position of the intersection point P.

4. Concluding remarks

In this paper, we have proposed a solution for the inverse kinematic problem of a 6-DOF manipulator under the condition that the orientation of the end-effector remains constant.

Our first task from here includes the verification of the solution with the CGS-QE method and efficiently solving the system of polynomial equations by using CGS, as we have proposed in the previous work. In addition, the orientation was specified this time for simplicity. However, orientation is not always constant in real-world manipulators. It is therefore necessary to develop the problem into an inverse kinematics problem for arbitrary orientations. There are several conditions on the geometry of the 6-DOF manipulator to be analytically solvable [13], and a method with computer algebra has been proposed for solving the inverse kinematic problem of the 6-DOF manipulator [3]. We will look for better solutions with reference to these methods.

References

[1] Elephant Robotics, Inc., 2024, mycobot 280, URL: https://www.elephantrobotics.com/en/mycobot-en/.

[2] M. P. Husty, Manfred L., H.-P. Schröcker, A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator, Mehcanism and machine theory 42.1 (2007) 66–81. doi:10.1016/j.mechmachtheory.2006.02.001.

[3] D. Manocha, J. Canny, Efficient inverse kinematics for general 6r manipulators, IEEE Transactions on Robotics and Automation 10 (1994) 648–657. doi:10.1109/70.326569.

[4] J.-C. Faugère, J.-P. Merlet, F. Rouillier, On solving the direct kinematics problem for parallel robots, Research Report RR-5923, INRIA, 2006. URL: https://hal.inria.fr/inria-00072366.

[5] C. M. Kalker-Kalkman, An implementation of Buchbergers' algorithm with applications to robotics, Mech. Mach. Theory 28 (1993) 523–537. doi:10.1016/0094-114X(93)90033-R.

[6] T. Uchida, J. McPhee, Triangularizing kinematic constraint equations using Gröbner bases for real-time dynamic simulation, Multibody System Dynamics 25 (2011) 335–356. doi:10.1007/s11044-010-9241-8.

[7] T. Uchida, J. McPhee, Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms, Mech. Mach. Theory 52 (2012) 144–157. doi:10.1016/j.mechmachtheory.2012.01.015.

[8] N. Horigome, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gröbner Bases, in: A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software – ICMS 2020, Springer International Publishing, Cham, 2020, pp. 3–13. doi:10.1007/978-3-030-52200-1_1.

[9] S. Otaki, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Real Quantifier Elimination based on Comprehensive Gröbner Systems, Preprint, 2021. doi:10.48550/arXiv.2111.00384, arXiv:2111.00384.

[10] M. Yoshizawa, A. Terui, M. Mikawa, Inverse Kinematics and Path Planning of Manipulator Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems, in: Computer Algebra in Scientific Computing: CASC 2023, volume 14139 of Lecture Notes in Computer Science, Springer, 2023, pp. 393–419. doi:10.1007/978-3-031-41724-5_21.

[11] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer, 2008. doi:10.1007/978-1-84628-642-1.

[12] A. A. Brandstötter, Mathias, M. Hofbaur, An analytical solution of the inverse kinematics problem of industrial serial manipulators with an ortho-parallel basis and a spherical wrist, in: Proceedings of the Austrian Robotics Workshop 2014, 2014, pp. 7–11.

[13] D. L. Pieper, The kinematics of manipulators under computer control, Stanford University, 1969. URL: https://apps.dtic.mil/sti/citations/AD0680036, accessed 2024-08-05, Ph.D. Thesis.

Methods for Solving the Post Correspondence Problem and Certificate Generation

Akihiro Omori[1], Yasuhiko Minamide[2]

[1]*Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan*
[2]*Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan*

Abstract
Post Correspondence Problem (PCP) is a well-known undecidable problem. Solving instances with solutions is straightforward with exploration algorithms, but proving infeasibility is challenging. This research introduces two methods to demonstrate infeasibility, including generating formal proofs in Isabelle/HOL.

Keywords
Post's Correspondence Problem, Formal Verification, Reachability Problem, Automata

1. Introduction

The Post Correspondence Problem (PCP), proposed by Post in 1946 [1], is undecidable. PCP instances use tiles with two strings on top and bottom.

100	0	1
1	100	00

In this example, there are three kinds of tiles, each available in infinite quantities. The problem is to determine whether it is possible to arrange one or more tiles in such a way that the reading of the top and bottom strings matches. In this particular instance, a solution (indices of arrangement of tiles) is "1311322", and this shows that both the top and bottom read "1001100100100".

100	1	100	100	1	0	0
1	00	1	1	00	100	100

For instances that have a solution, it is possible to find the solution within finite time using an exploration algorithm. On the other hand, determining that no solution exists is challenging, and due to the undecidability of the problem, no general algorithm exists for this purpose. Previous research has proposed heuristic algorithms for finding solutions [2, 3] and Ling Zhao (2003) [2] attempted to solve all the problems in *PCP[3,4]* and left 3,170 problems unsolved. *PCP[3,4]* refers to a set of all instances where the number of tiles is 3, and the maximum length of the written strings is 4.

This research makes the following three main contributions.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
omori.a.ab@m.titech.ac.jp (A. Omori); minamide@c.titech.ac.jp (Y. Minamide)

- Propose two novel algorithms to demonstrate that a PCP instance has no solution.
- Solve all problems of *PCP[3,4]* except for 13 problems.
- Show an example of automatic proof generation for concrete problems.

2. The First Method: String Constraint Formulation

We formulate PCP as a string constraint problem.

Example 2.1 (Example of T_g and T_h)**.** Let PCP instance $I = ((1111, 1110), (1101, 1), (11, 1111))$. We denote the top and bottom strings on the i-th tile by g_i and h_i, respectively. Let T_g and T_h be transducers as defined below. Intuitively, the transducer T_g outputs g_1 for '1', g_2 for '2', and does not accept the empty string. The string w is a solution to the PCP if and only if $T_g(w) = T_h(w)$.

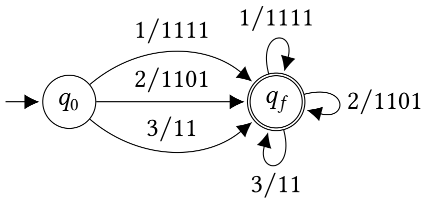


Figure 1: T_g

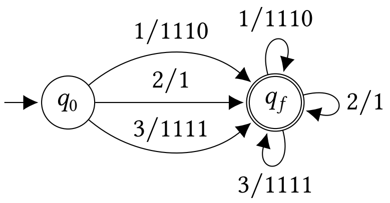


Figure 2: T_h

Definition 2.2 (String Constraint of PCP)**.** The constraint $T_g(w) = T_h(w)$ created in this way is called the string constraint of instance *I*.

Regarding the string constraint ϕ of the PCP instance *I*, the satisfiability is undecidable. We consider ϕ' such that $\phi(w) \implies \phi'(w)$ and is efficiently decidable. Such ϕ' is referred to as a relaxation problem (simply relaxation) of ϕ. By showing that ϕ' is unsatisfiable, we would like to show that ϕ is also unsatisfiable. For example, considering only the number of characters $|T_g(w)| = |T_h(w)|$ is suitable as ϕ'. Additionally, matching Parikh images or the number of specific words is another example of ϕ'. Generalizing these examples, we have the following proposition.

Proposition 2.3. Let W be an arbitrary total integer-vector-output transducer. Consider

$$W(T_g(w)) \cap W(T_h(w)) \neq \emptyset \tag{1}$$

This is a relaxation problem of ϕ and is decidable. We set some W and hope it is infeasible.

Although details are omitted, the condition $W(T_g(w)) \cap W(T_h(w)) \neq \emptyset$ can be reduced to the emptiness problem of a Parikh automaton constructed from W, T_g, T_h, and their product. The Parikh automaton emptiness algorithm we use is largely similar to the one described in Section 3 of [4], so we omit the details. While not detailed here, our algorithm achieved significant speedup by applying two techniques to this algorithm: (1) delaying and dynamically adding constraints related to connectivity, and (2) reducing the problem to a natural form for Mixed Integer Programming and leveraging a cutting-edge MIP solver.

3. The Second Method: Transition System Formulation

Intuitively, arranging each tile one by one represents a transition, and “the remaining part of the string and whether it is on the top or bottom” represents a state. We call such a pair *configuration.* PCP can be formulated as a reachability problem: “Is it possible to reach the state of the empty string?”

Example 3.1. When arranging two tiles like $\begin{array}{|c|}\hline 100\\ \hdashline 1\\ \hline\end{array}\begin{array}{|c|}\hline 10\\ \hdashline 0\\ \hline\end{array}$, the state representing it is “top, remainder 010.” If a transition is made by appending $\begin{array}{|c|}\hline 111\\ \hdashline 01\\ \hline\end{array}$, the next state will be “top, remainder 0111.”

3.1. Problem Definition

We formulate PCP as a reachability problem. First, we define the transition system of PCP.

Definition 3.2 (Transition System of PCP)**.** Let $I = ((g_1, h_1), \dots, (g_s, h_s))$ be a PCP instance of size s over Σ. We define the transition system $Tr = (Q, T, Init, Bad)$ as follows.

- State set $Q = \{\text{top}, \text{bottom}\} \times \Sigma^*$.
- Transition function $T : Q \to 2^Q$ is defined as follows.

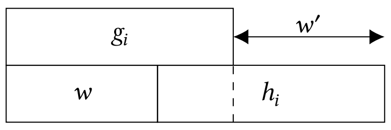
$$T(\text{bottom}, w) = \{(\text{bottom}, w') \mid \exists i \leq s.\ wh_i = g_i w'\} \cup \{(\text{top}, w') \mid \exists i \leq s.\ wh_i w' = g_i\}$$

$$T(\text{top}, w) = \{(\text{top}, w') \mid \exists i \leq s.\ wg_i = h_i w'\} \cup \{(\text{bottom}, w') \mid \exists i \leq s.\ wg_i w' = h_i\}$$

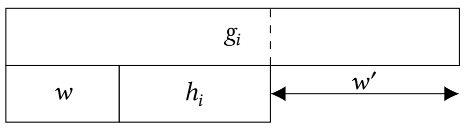
- Bad state set $Bad = \{(\text{top}, \epsilon), (\text{bottom}, \epsilon)\}$.
- Intial state set $Init = T(\text{top}, \epsilon)$.
  The states after arranging one tile is considered the initial state, as an empty arrangement is not valid.

In the following, T is naturally extended and used as $T : 2^Q \to 2^Q$.

The behavior of the transition $T(\text{bottom}, w)$ is illustrated below. When w is the current state, adding (g_i, h_i) results in the remaining part becoming the next state w' . There are two patterns: one where the same side as the previous state continues, and one where the side changes.



(a) Pattern where the side doesn’t change



(b) Pattern where the side changes

Definition 3.3 (Reachability Problem of PCP)**.** Does there exist n such that

$$T^n(Init) \cap Bad \neq \emptyset$$

Definition 3.4 (Inductive Invariant of PCP)**.** A set *Inv* that satisfies the following three conditions is called an inductive invariant (simply invariant).

- $\mathit{Init} \subseteq \mathit{Inv}$
- *Inv* is closed under *T*: $T(\mathit{Inv}) \subseteq \mathit{Inv}$
- *Inv* does not include ϵ: $\mathit{Bad} \cap \mathit{Inv} = \emptyset$

Lemma 3.5. If *Inv* exists, then it implies that *Bad* is unreachable from initial states.

In the following section, we introduce algorithms to discover *Inv*.

3.2. Algorithm

For the Reachability Problem, many powerful algorithms like PDR (Property Directed Reachability)[5] exist. We extended PDR and achieved some success (see Section 5). We also devised a novel ad-hoc method specific to PCP, described below.

Definition 3.6 (Configuration Automaton)**.** Let $s \in \{\text{top}, \text{bottom}\}$ and A be a finite automaton over Σ. We call the pair (s, A) the configuration automaton. The language of (s, A) is denoted as $L(s, A)$ and defined as follows. This represents a state set of the transition system.

$$L(s, A) = \{(s, w) \mid w \in L(A)\}$$

The aim of this algorithm is to discover a pair of configuration automata (for top and bottom) that represents *Inv*. It should be noted that not every *Inv* has such a pair due to the regularity of the underlying automata, which limits the scope of our consideration.

This algorithm manages a graph $G = (V, E)$ where each node is a configuration automaton. Specifically, each node v is associated with a set of states of a transition system. The algorithm proceeds by expanding the overall union $L(V) = \bigcup_v L(v)$ until it becomes an invariant.

Intuitively, the edge (u, v) in this graph represents a dependency relationship. This relationship means "if v cannot reach *Bad*, then u cannot reach *Bad* either". If we can construct a graph where every node has such dependencies and does not contain any bad state, then $L(V)$ is an invariant. There are two types of this relation, as follows.

1. Inclusion relation: $L(u) \subsetneq L(v)$
2. Transition relation: $T(L(u)) = L(v)$

The algorithm is essentially a breadth-first search (BFS). When considering only the transition relation, the process operates similarly to BFS. A distinctive feature of this algorithm is that it proactively *abstracts* nodes. For example, when a node such as $(\text{top}, 0011101)$ appears, the algorithm attempts to create a node like $(\text{top}, .{*}110.{*})$ (we use a regex to represent an automaton) and draw an edge to it. If this abstracted node can reach *Bad*, it is removed and backtracking is performed.

Figure 4 shows a successful execution example for $\boxed{\begin{array}{c}1111\\ \hline 1\end{array}}\ \boxed{\begin{array}{c}0\\ \hline 11\end{array}}\ \boxed{\begin{array}{c}1\\ \hline 1100\end{array}}$. The square nodes represent nodes with singleton languages, and the round nodes are abstracted nodes with regular expressions appearing in their labels. The dotted lines represent inclusion relations, and the solid lines represent transition relations. Note that in this figure, the transition relations are extended to n (where $n \geq 1$) steps, with intermediate steps omitted.

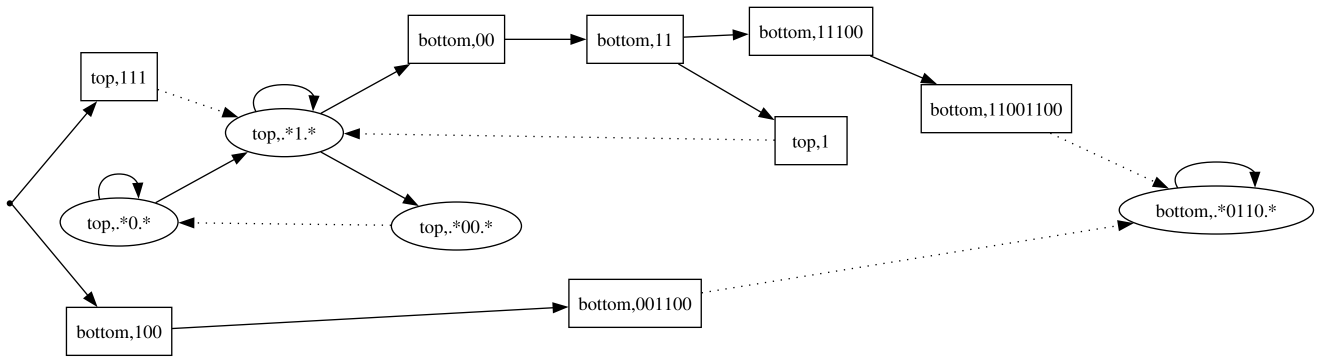


Figure 4: Example of Graph of Invariant

4. Certificate Generation

So far, we have presented two methods and complicated algorithms. However, there is a significant possibility that my implementations for these algorithms may contain bugs. Even if we successfully solve all instances of *PCP[3,4]*, our results would still be far from being considered trusted facts. Therefore, we decided to have our algorithm output proofs in the form of Isabelle/HOL code.

Another possible approach is to use Isabelle/HOL or similar tools to verify the correctness of the algorithm's implementation. However, this makes it difficult to optimize the algorithm for speed. For instance, The first method relies on an external MIP solver for its efficiency, making it challenging. Additionally, for others to quickly trust our results, it is crucial that all instances of $PCP[3,4]$ and their proofs are organized and verified within some proof assistant such as Isabelle/HOL.

Currently, only the second method is capable of outputting a certificate. The first method will be addressed as future work (see Section 6).

4.1. Certificate: Pair of Automata

Consider the transition system of a PCP instance. By defining the invariant concretely in Isabelle/HOL and proving each of the invariant conditions (see Definition 3.4), we can validate it. This method is independent of the implementation details used in the second method and can be utilized by various algorithms discovering invariants.

Our implementation of the second method generates the following code.

1. Definition of the PCP instance
2. Definition of Inv
   a) The top-side Automaton
   b) The bottom-side Automaton
3. Proof of the closedness of Inv
   a) Definition of $T(Inv)$ (in the form of a specific pair of deterministic automata)

b) Concrete definition of the automaton for $Inv \cap \overline{T(Inv)}$
c) Proof of $Inv \cap \overline{T(Inv)} = \emptyset$
d) Proof of $\overline{Inv} \cap T(Inv)$ similarly, and show that $T(Inv) \subseteq Inv$

Proofs such as "the existence of *Inv* implies that the PCP has no solution" were conducted manually in advance. Examples of complete proofs are found on the author's GitHub repository [6].

5. Application to *PCP[3,4]*

In this research, we address the instances of *PCP[3,4]*. Ling Zhao (2003) [2] attempted to solve all these instances but left 3,170 unsolved. The list of these instances is available on his website [7]. Our goal was to solve all instances of *PCP[3,4]*, gradually reducing the number of unsolved problems. As shown in Figure 5, the initial 3,170 unsolved problems were reduced to 127 using the first method. After several additional methods, only 13 problems remained unsolved. These remaining problems are listed on the author's website [8].

PDR, SAT, Method2(1), and Method2(2) are techniques for discovering *Inv*. Certificate generation is implemented for those methods. The method SAT uses a SAT solver to discover *Inv*, while Method2(1) and Method2(2) differ in their abstraction methods.

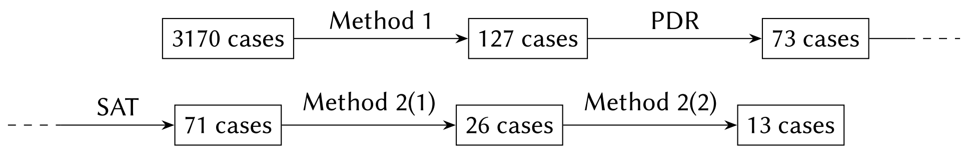


Figure 5: Journey of Solving *PCP[3,4]*

6. Conclusion and Future Work

We have been working for a complete resolution of *PCP[3,4]* and came close, with only 13 instances remaining unsolved. To have these results accepted as trusted facts, we also aim to provide formal proofs using Isabelle/HOL for each instance, which has been achieved for the second method. Although both goals are yet to be fully achieved, we believe they are attainable as outlined below.

To solve the remaining 13 problems, we consider two possibilities. One is to solve these instances manually. We predict that most of the 13 problems do not have solutions, and providing ad-hoc proofs by humans might be the quickest way. The other possibility involves devising new variants of the methods in this paper or investing additional computational resources. Since the manual approach can also help gain deeper insights into individual instances and PCP itself, we would like to first aim for manual resolution.

Generating certificates for the first method is challenging because it uses an external Mixed Integer Programming (MIP) solver as a subroutine. Generating a certificate for the feasibility of an MIP is straightforward, as it merely requires providing a specific solution. However, generating a certificate for infeasibility is more difficult. Cheung et al. (2017) [9] extended the existing MIP solver SCIP to output easily verifiable certificates in their own format. We believe that we can overcome this difficulty by converting these certificates into Isabelle/HOL code.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 19K11899 and 24K14891.

References

[1] E. L. Post, A variant of a recursively unsolvable problem, Bulletin of the American Mathematical Society 52 (1946) 264–268.

[2] L. Zhao, Tackling Post's correspondence problem, in: Computers and Games, Springer Berlin Heidelberg, 2003, pp. 326–344.

[3] R. J. Lorentz, Creating difficult instances of the post correspondence problem, in: Computers and Games, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 214–228.

[4] N. Verma, H. Seidl, T. Schwentick, On the complexity of equational horn clauses, 2005, pp. 337–352. doi:10.1007/11532231_25.

[5] A. R. Bradley, Sat-based model checking without unrolling, in: Proceedings of the 12th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI'11, Springer-Verlag, Berlin, Heidelberg, 2011, p. 70–87.

[6] A. Omori, pcp-proof, https://github.com/Mojashi/pcp-proof, 2024.

[7] L. Zhao, Pcp documents, 2002. URL: https://webdocs.cs.ualberta.ca/~games/PCP.

[8] A. Omori, Unresolved problems, 2024. URL: https://pcp-vis.pages.dev/gallery.

[9] K. K. H. Cheung, A. Gleixner, D. E. Steffy, Verifying integer programming results, in: F. Eisenbrand, J. Koenemann (Eds.), Integer Programming and Combinatorial Optimization, Springer International Publishing, Cham, 2017, pp. 148–160.

A Stable Computation of Multivariarte Apporximate GCD Based on SVD and Lifting Technique

Masaru Sanuki[1,*,†]

[1]Institute of Medicine, University of Tsukuba, Ten-noudai 1-1-1, Tsukuba-shi, Ibaraki 305-8575, Japan

Abstract

For univariate polynomials, the approximate GCD can be obtained by computing the null space of the subresultant matrix of given polynomials. In this study, for multivariate polynomials, we propose a method for computing null space of the subresultant matrix within polynomials stably and efficiently, which is based on the SVD (singular value decomposition) and lifting techniques. Therefore, we show the multivariate approximate GCD can be also computed by using subresultant matrix. In addition, we describe an ill-conditioned case (initial factors have approximate common factor) and solve them.

Keywords

Approximate GCD, Lifting technique, Ill-conditioned cases

1. Preliminaries

Let $F(x, \boldsymbol{t})$ and $G(x, \boldsymbol{t})$ be multivariate polynomials in $\mathbb{F}[x, t_1, \ldots, t_\ell] = \mathbb{F}[x, \boldsymbol{t}]$ (x is the main variable and $\boldsymbol{t} = (t_1, \ldots, t_\ell)$ are sub-variables), and be expressed as

$$\begin{aligned} F(x, \boldsymbol{t}) &= \tilde{F}(x, \boldsymbol{t})C(x, \boldsymbol{t}) + \Delta_F = f_m(\boldsymbol{t})x^m + \ldots + f_0(\boldsymbol{t}), \\ G(x, \boldsymbol{t}) &= \tilde{G}(x, \boldsymbol{t})C(x, \boldsymbol{t}) + \Delta_G = g_n(\boldsymbol{t})x^n + \ldots + g_0(\boldsymbol{t}). \end{aligned}$$

Here, $\tilde{F}, \tilde{G}, C, \Delta_F, \Delta_G$ are polynomials in $\mathbb{F}[x, \boldsymbol{t}]$, and when $||\Delta_F|| \ll ||F||$ and $||\Delta_G|| \ll ||G||$, C is called an approximate factor of F and G. In particular, the approximate factor of maximum degree is called approximate GCD, which is denoted by $\gcd(F, G)$.

Various algorithm exist for the approximate GCD of univariate polynomials. However, there are few stable all-purpose methods for a large number of variables in a multivariate case. Numerical-based methods are stable but significantly less efficient, so we have tried to improve efficiency by combining lifting methods [4]. In this study, we challenge the stable computation of the null space of the subresultant within polynomial entries.

First, we review the method for the subresultant matrix for multivariate polynomials. For the resultant within polynomials, we propose a QRGCD-like method over truncated power-series polynomials, it is efficient [5]. For the null space of the subresultant matrix, Gao *et al.* and Zeng-Dayton proposed SVD-based methods for numeric matrices at the same conference [2, 7], where the SVD is the *singular value decomposition* for matrix. These matrices are sparse and the size are also huge extremely although the degree of given polynomial is not large. Lifting techniques is known for solving of equation modulo an ideal I and lifting them to solution modulo $I^2, I^3, \ldots$ in order to get the ideal adic completion. Here I is an ideal as $I = \langle t_1 - s_1, \ldots, t_\ell - s_\ell \rangle$ with $(s_1, \ldots, s_\ell) \in \mathbb{F}^\ell$ (in this paper, $(s_1, \ldots, s_\ell)$ is the origin). For multivariate GCD computation, the EZ-GCD method is well-known lifting method based on Hensel's lemma, however, its approximate computation will be unstable when initial factors have an approximate common factor [8].

In this paper, we propose a stable multivariate approximate GCD computation, which is based on the SVD and lifting techniques. It is able to compute the approximate GCD even though initial factors have approximate common factors.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
*Corresponding author.
sanuki@md.tsukuba.ac.jp (M. Sanuki)

2. Framework of algorithm

In this paper, we discuss non-singular case only, i.e., $F(x, 0) \cdot G(x, 0) \neq 0$ and $f_m(0) \cdot g_n(0) \neq 0$. For non-singular case, every polynomial $P(x, \boldsymbol{t})$ is transform to $P(x, T, \boldsymbol{t}) = P(x, Tt_1, \ldots, Tt_\ell)$, with T is the total-degree variable. Every polynomial $P(x, T, \boldsymbol{t})$ is represented as the sum of homogeneous polynomials w.r.t. the total-degree variable T;

$$\begin{aligned} P(x, T, \boldsymbol{t}) &= P^{(0)}(x) + T \cdot \delta P^{(1)}(x, \boldsymbol{t}) + \ldots + T^w \cdot \delta P^{(w)}(x, \boldsymbol{t}) + \ldots, \\ P^{(w)}(x, T, \boldsymbol{t}) &= P^{(0)}(x) + T \cdot \delta P^{(1)}(x, \boldsymbol{t}) + \ldots + T^w \cdot \delta P^{(w)}(x, \boldsymbol{t}). \end{aligned}$$

In non-singular case, the following two conditions exist: $\deg(\gcd(F, G)) \leq \deg(\gcd(F^{(0)}, G^{(0)}))$ and $\gcd(F, G) | \gcd(F^{(0)}, G^{(0)})$. In such situations, lifting algorithms can be applied. The proposed algorithm is discussed in the next section.

2.1. Computing cofactors of F and G via lifting method

Let $\mathcal{S}_i(F, G) = \mathcal{S}_i \in \mathbb{F}[\boldsymbol{t}, T]^{(m+n-2i)\times(m+n-2i)}$ be an ith-subresultant matrix of F and G w.r.t. x, and be represented as

$$\begin{aligned} \mathcal{S}_i &= \begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_{m-n-k} & & f_m & g_{n-m-k} & & g_n \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ & & f_{m-n-k} & & & g_{n-m-k} \end{pmatrix} \\ &= \mathcal{S}_i^{(0)} + T \cdot \delta\mathcal{S}_i^{(1)} + \ldots + T^w \cdot \delta\mathcal{S}_i^{(w)} + \ldots, \end{aligned}$$

where $\mathcal{S}_i^{(0)} = \delta\mathcal{S}_i^{(0)} \in \mathbb{F}^{(m+n-2i)\times(m+n-2i)}$ and $\delta\mathcal{S}_i^{(w)} \in \mathbb{F}[\boldsymbol{t}]^{(m+n-2i)\times(m+n-2i)}$ for $w \geq 1$.

When $k = \deg(\gcd(F, G)) = \deg(\gcd(F^{(0)}, G^{(0)}))$, it is well-known as the null space of $\mathcal{S}_{k-1}$ corresponds to $\tilde{G}$ and $\tilde{F}$, and $\operatorname{rank}(\mathcal{S}_{k-1}) = K - 1$ where $K = m - (k-1) + n - (k-1)$.

computation of cofactors for univariate part: SVD

Cofactors of $F^{(0)}$ and $G^{(0)}$ can be obtained from the null space of $\mathcal{S}_{k-1}^{(0)}$. In this paper, we compute the null space of $\mathcal{S}_{k-1}^{(0)}$ using the SVD [1]. Using the SVD of $\mathcal{S}_{k-1}^{(0)}$, we obtain the following decomposition:

$$\mathcal{S}_{k-1}^{(0)} = U\Sigma V^T = \begin{pmatrix} \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_K \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_K \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_K^T \end{pmatrix},$$

where $K = m - (k-1) + n - (k-1)$, U and V are orthogonal matrices, and σ_i are singular vectors with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{K-1} \gg \sigma_K \geq 0$, respectively. Then, $\boldsymbol{v}_K \in \operatorname{Ker}(\mathcal{S}_{k-1}^{(0)})$, and it is one of the solutions of $\mathcal{S}_{k-1}^{(0}\boldsymbol{z} = \boldsymbol{0}$ is $\boldsymbol{z} = \boldsymbol{r}^{(0)} = \boldsymbol{v}_K$;

$$\boldsymbol{v}_K = \begin{pmatrix} \tilde{g}_{n-k}^{(0)} \\ \\ \tilde{g}_0^{(0)} \\ \hline -\tilde{f}_{m-k}^{(0)} \\ \\ -\tilde{f}_0^{(0)} \end{pmatrix}, \text{ with } ||\boldsymbol{v}_K||_2 = 1.$$

Computation of cofactors for multivariate part: lifting method

Suppose we have $\boldsymbol{r}^{(w)} = \boldsymbol{r}^{(w-1)} + \delta\boldsymbol{r}^{(w)}$. Here, $\delta\boldsymbol{r}^{(w)}$ is a vector generated by homogenious polynomials with total-degree w w.r.t. T. Then, $\delta\boldsymbol{r}^{(w+1)}$ is generated as follows. Note that the following consists.

$$\begin{aligned}\mathcal{S}_{k-1}\boldsymbol{r}^{(w+1)} &\equiv \boldsymbol{0} \pmod{T^{w+2}}\\ \mathcal{S}_{k-1}^{(0)}\delta\boldsymbol{r}^{(w+1)} &= -\sum_{j=1}^{w+1}\mathcal{S}_{k-1}^{(j)}\delta\boldsymbol{r}^{(w-j)} = \delta\boldsymbol{p}^{(w)}.\end{aligned}$$

Now, $\delta\boldsymbol{r}^{(w+1)}$ and $\delta\boldsymbol{p}^{(w+1)}$ are transformed bases from $\boldsymbol{e}_1,\ldots,\boldsymbol{e}_K$ to $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_K$ and $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_K$, respectively, as follows:

$$\delta\boldsymbol{z}^{(w)} = \begin{pmatrix}\delta z_1^{(w)}\\ \vdots\\ \delta z_K^{(w)}\end{pmatrix} = \delta\hat{z}_1^{(w)}\boldsymbol{v}_1 + \ldots + \delta\hat{z}_K^{(w)}\boldsymbol{v}_K,\quad \delta\boldsymbol{p}^{(w)} = \begin{pmatrix}\delta p_1^{(w)}\\ \vdots\\ \delta p_K^{(w)}\end{pmatrix} = \delta\hat{p}_1^{(w)}\boldsymbol{u}_1 + \ldots + \delta\hat{p}_N^{(w)}\boldsymbol{u}_K.$$

Then, we obtain $\delta\hat{z}_i^{(w+1)} = \delta\hat{p}_i^{(w+1)}/\sigma_i \quad (i = 1,\ldots,K-1)$. Therefore, $\delta\boldsymbol{r}^{(w+1)}$ is constructed, as follows.

$$\delta\boldsymbol{r}^{(w+1)} = \delta\hat{p}_1^{(1)}/\sigma_1\boldsymbol{v}_1 + \ldots + \delta\hat{p}_{K-1}^{(1)}/\sigma_{K-1}\boldsymbol{v}_{K-1} + \mathbb{F}[T,\boldsymbol{t}]_{w+1}\cdot\boldsymbol{v}_K,$$

where $\mathbb{F}[T,\boldsymbol{t}]_{w+1}$ is homogeneous polynomial set with total-degree $w+1$ w.r.t. T, and we have the following as a candidate solution.

$$\boldsymbol{r}^{(w)} = \boldsymbol{t}_K + \sum_{j=1}^{w}\delta\hat{p}_1^{(w)}/\sigma_1\boldsymbol{v}_1 + \ldots + \sum_{j=1}^{w}\delta\hat{p}_{K-1}^{(w)}/\sigma_{K-1}\boldsymbol{v}_{K-1} + \mathbb{F}[\boldsymbol{T},\boldsymbol{t}]_{[1,w]}\cdot\boldsymbol{v}_K,$$

where $\mathbb{F}[\boldsymbol{T},\boldsymbol{t}]_{[1,w]} = \cup_{j=1}^{w}\mathbb{F}[T,\boldsymbol{t}]_j$. To compute the approximate GCD of F and G, we need to determine $q^{(w)} = \delta q^{(1)} + \ldots + \delta q_{(w)} \in \mathbb{F}[\boldsymbol{T},\boldsymbol{t}]_{[1,w]}$ s.t.

$$\boldsymbol{r}^{(w)}(q) = \boldsymbol{t}_K + \sum_{j=1}^{w}\delta\hat{p}_1^{(w)}/\sigma_1\boldsymbol{v}_1 + \ldots + \sum_{j=1}^{w}\delta\hat{p}_{K-1}^{(w)}/\sigma_{K-1}\boldsymbol{v}_{K-1} + q^{(w)}\cdot\boldsymbol{v}_K,$$

To determine the approximate GCD, we must determine $q^{(i)}$. The following example shows one approach to determining each undetermined element $\delta q^{(i)}$ for $1 \le i \le w$..

Example1

Polynomials $F(x,t_1,t_2)$ and $G(x,t_1,t_2)$ having an approximate GCD $C(x,t_1,t_2) = x^3 + (1 + t_2 - 2t_1 + {t_1}^2)x + 3$ are expressed as

$$\begin{aligned}F(x,t_1,t_2) &= (x^3 + ({t_2}^2 + t_1 + t_1 - 2)x^2 - 1)\times C(x,t_1,t_2) + M_{eps},\\ G(x,t_1,t_2) &= (x^3 + (2{t_2}^2 - t_1 + 3)x^2 - 1)\times C(x,t_1,t_2) + M_{eps},\end{aligned}$$

where M_{eps} is the machine epsilon.

In this example, $k = 2$ is already known ($K = 8$). Then, one solution of $\mathcal{S}_1^{(0)}\boldsymbol{z} = \boldsymbol{0}$ is $\boldsymbol{z} = \boldsymbol{r}^{(0)} = \boldsymbol{v}_8$;

$$\boldsymbol{r}^{(0)} = \boldsymbol{v}_8 = \left(\begin{array}{c}-0.242535625036333\\ -0.727606875108999\\ -2.24840273230668\times 10^{-15}\\ 0.242535625036333\\ \hline 0.242535625036333\\ -0.485071250072665\\ -1.32375311946987\times 10^{-15}\\ -0.242535625036333\end{array}\right).$$

A candidate of $\delta \boldsymbol{r}^{(1)}|_{\delta q^{(1)}=0} = \delta \hat{p}_1^{(1)}/\sigma_1 \boldsymbol{v}_1 + \ldots + \delta \hat{p}_7^{(1)}/\sigma_7 \boldsymbol{v}_7 + \delta q^{(1)} \times \boldsymbol{v}_8$ is

$$\delta \boldsymbol{z}^{(1)} = \left(\begin{array}{c} 0.0713340073\cdots t_1 + 0.0285336029\cdots t_2 \\ -0.0285336029\cdots t_1 + 0.0856008088\cdots t_2 \\ 3.65419500\cdots \times 10^{-15} t_1 - 4.96824803\cdots \times 10^{-15} t_2 \\ -0.0713340073\cdots t_1 - 0.0285336029\cdots t_2 \\ \hline -0.0713340073\cdots t_1 - 0.0285336029\cdots t_2 \\ -0.0998676103\cdots t_1 - 0.185468419\cdots t_2 \\ 4.77577504\cdots \times 10^{-16} t_1 - 1.10469359\cdots \times 10^{-15} t_2 \\ 0.0713340073\cdots t_1 + 0.0285336029\cdots t_2 \end{array}\right) + \delta q^{(1)} \cdot \boldsymbol{v}_8 = \left(\begin{array}{c} \delta \tilde{g}_{n-k}^{(1)} \\ \delta \tilde{g}_{n-k-1}^{(1)} \\ \vdots \\ \delta \tilde{g}_0^{(1)} \\ \hline -\delta f_{m-k}^{(1)} \\ -\delta f_{m-k-1}^{(1)} \\ \vdots \\ -\delta f_0^{(1)} \end{array}\right).$$

Generally, it is difficult to determine $\delta q^{(1)}$ properly.

However, assuming that cofactors are also not dense or the approximate GCD is monic, several coefficients will be zero. In this example, assume the 1st element is zero, $\delta q^{(1)}$ is $\delta q_1^{(1)} = (0.0713340073636269\, t_1 + 0.0285336029454512\, t_2)/0.242535625036333$ and $\delta \boldsymbol{r}^{(1)}$ becomes

$$\left(\begin{array}{c} 0 \\ 0.242535625036331 t_1 \\ 0 \\ 0 \\ \hline 0 \\ -0.242535625036331 t_1 - 0.242535625036334 t_2 \\ 0 \\ 0 \end{array}\right).$$

It is unlikely that many factors will be close to zero simultaneously, and this can only happen if the result is correct. Unlike the EZ-GCD method, it is more efficient because it can extract each undetermined coefficient at each lifting step. So that, "check zeros" is very efficiency.

If the coefficients are dense, $\mathrm{lc}(\mathrm{lc}(F), \mathrm{lc}(G))$ or $\mathrm{lc}(\mathrm{lc}(\gcd(F, G)))$ should be calculated in advance so that the elements can be determined uniquely.

2.2. Computing approximate GCD

After obtaining cofactors, the approximate GCD is computed by solving and $\boldsymbol{F} = (f_m, f_{m-1}, \ldots, f_0)^T \in \mathbb{F}[\boldsymbol{t}]^{m+1}$.

$$\begin{pmatrix} \tilde{f}_{m-k} & & \\ \vdots & \ddots & \\ \tilde{f}_0 & & \tilde{f}_{m-k} \\ & \ddots & \vdots \\ & & \tilde{f}_0 \end{pmatrix} \begin{pmatrix} c_k \\ c_{k-1} \\ \vdots \\ c_0 \end{pmatrix} = \begin{pmatrix} f_m \\ f_{m-1} \\ \vdots \\ f_0 \end{pmatrix}$$

This linear equation is solve as following step.

1. Solve $\mathcal{C}_{m+1,k+1}^{(0)}(\tilde{\boldsymbol{F}}) \cdot \boldsymbol{c}^{(0)} = \boldsymbol{F}^{(0)}$. Actually, we utilize the SVD as in the former case.
2. Lifting step: solve $\mathcal{C}_{m+1,k+1}^{(0)}(\tilde{\boldsymbol{F}}) \cdot \delta \boldsymbol{c}^{(w)} = \delta \boldsymbol{F}^{(w)} - \sum_i C_{m+1,k+1}^{(i)}(\tilde{\boldsymbol{F}}) \times \delta \boldsymbol{c}^{(w-i)}$. This step can also be solved using SVD.
3. Return $c_k x^k + \cdots + c_0$ as an approximate GCD.

3. Solve in ill-conditioned cases

In this section, we demonstrate that our method is stable for ill-conditioned cases [8, 5]. On the other word, we deals with cases where the initial factor is an approximate common factor. In this case, the EZ-GCD method is unstable since large cancellation errors occur [6].

Example 2 (initial factors have approximate common factor)

Compute the approximate GCD of F and G, where both polynomials are monic.

$$\begin{aligned}
F(x,t_1,t_2) &= (x^3+({t_2}^2+t_1+t_2-2)x^2-1)(x-1.0003+2t_2-{t_1}^2)C+M_{eps}, \\
G(x,t_1,t_2) &= (x^3+(2{t_2}^2-t_1+3)x^2-1)(x-1.0005+t_1+t_2+t_1t_2)C+M_{eps}, \\
C(x,t_1,t_2) &= x^3+(1+t_2-2t_1+{t_1}^2)x+3.
\end{aligned}$$

Initial factors $F^{(0)}$ and $G^{(0)}$ have an approximate common factor $(x-1.0002)$ with tolerance $O(10^{-5})$. Sigular values of $\mathcal{S}_3^{(0)}(F,G)$ are $19.8 > 18.3 > 14.5 > 12.8 > 8.2 > 4.4 > 1.1 > 0.6 \gg 1.5\times10^{-5} \gg 1.1\times10^{-16}$. Because give polynomials are monic, the leading coefficient of cofactors and the approximate GCD are also monic, respectively.

When $w=1$, Adjusting the 1st element of $\delta\boldsymbol{r}^{(1)}$ by $\boldsymbol{z}_K$ only, we obtained the following.

$$\delta\boldsymbol{r}^{(1)} = \begin{pmatrix}
0. \\
-7.25814180424500\times10^{-8}t_1+0.176741414005228t_2 \\
0.707053518826616t_1+0.530224242015704t_2 \\
-6.96526170074208\times10^{-14}t_1+6.29774010718620\times10^{-14}t_2 \\
-0.176741268890038t_1-0.176741414005196t_2 \\
1.42941214420489\times10^{-15}t_1+6.38378239159465\times10^{-16}t_2 \\
-0.176741268889989t_1-0.530224096948041t_2 \\
0.176794218725546t_1+0.883759874841642t_2 \\
-1.18932641512970\times10^{-14}t_1-7.57727214306669\times10^{-15}t_2 \\
-7.25814328778052\times10^{-8}t_1+0.353482755476628t_2
\end{pmatrix}$$

The perturbation is $||\tilde{F}^{(1)}G^{(1)}-\tilde{F}^{(1)}G^{(1)}|| \approx O(10^{-8}) \approx \sigma_K/\sigma_{K-1}$. On the other hand, by adjusting $\boldsymbol{z}_{K-1}$, we obtained the following, It can be confirmed that the solution is not accurate; $||\tilde{F}^{(1)}G^{(1)}-\tilde{F}^{(1)}G^{(1)}|| \approx \sigma_K$.

$$\begin{pmatrix}
0. \\
-0.19885118257931 07t_1-0.14357803747786974t_2 \\
0.11055165805122452t_1-0.43065120320683287t_2 \\
0.0002976768351589665t_1+0.0004795129410803 4004t_2 \\
0.02231804334219 7766t_1+0.14391342019466513t_2 \\
-0.7183155936049679\times10^{-5}t_1-0.000011570991832604571t_2 \\
0.02212670015656562t_1-0.20987748805445672t_2 \\
-0.22096310056080598t_1+0.243032215144183t_2 \\
0.00007010876634662433t_1+0.00011293475597320968t_2 \\
-0.19880976332099576t_1+0.03323002423516758t_2
\end{pmatrix}$$

When $w=2$, by lifting step and adjusting the 1st element of $\delta\boldsymbol{r}^{(2)}$ by $\boldsymbol{z}_K$, we obtained the following vector.

$$\left(\begin{array}{c} 0. \\ 0.353120013467623{t_2}^2 + 0.177466845429488t_1t_2 - 0.000362979823316206{t_1}^2 \\ -0.354747432749536{t_2}^2 + 0.355659122269873t_1t_2 - 0.177830208344029{t_1}^2 \\ 8.84819995\cdots\times 10^{-13}{t_2}^2 - 4.59634934\cdots\times 10^{-12}t_1t_2 + 1.03052288\cdots\times 10^{-12}{t_1}^2 \\ 0.000362669479650586{t_2}^2 - 0.177466845422696t_1t_2 + 0.000362979818887450{t_1}^2 \\ \hline -9.08967345\cdots\times 10^{-13}{t_2}^2 - 3.63698827\cdots\times 10^{-12}t_1t_2 - 1.36436695\cdots\times 10^{-12}{t_1}^2 \\ -0.176378671991595{t_2}^2 - 0.000725503949587463t_1t_2 + 0.177104321289445{t_1}^2 \\ -0.177413730546595{t_2}^2 - 0.352031674994445t_1t_2 - 0.354208569999507{t_1}^2 \\ -9.15635622\cdots\times 10^{-13}{t_2}^2 + 2.71640175\cdots\times 10^{-12}t_1t_2 - 1.68760838\cdots\times 10^{-13}{t_1}^2 \\ -0.000362669478412958{t_2}^2 + 0.000725503952765407t_1t_2 - 0.177104321291319{t_1}^2 \end{array}\right)$$

Adjusting only $\boldsymbol{v}_K$ is not accurate. Therefore, adjusting $\boldsymbol{v}_K$ and $\boldsymbol{v}_{K-1}$ in $\ker \mathcal{S}_k^{(0)}$ we have the following, and it obtains the expected solution one . In this case, perturbation becomes $||\tilde{F}^{(2)}G^{(2)} - \tilde{F}^{(2)}G^{(2)}|| \approx \sigma_K$, it is better.

The SVD is stable even if the matrix is irregular. Thus, the SVD of $\mathcal{S}^{(0)}$ is also stable even if initial factors have an approximate common factor. On the other hand, a lifting method using the Bezout matrix is unstable since initial matrix is assumed to be regular [4]. Hence, our method is more stable and efficient than existing methods.

References

[1] R. Corless, P. Gianni, B. Trager and S. Watt, *The singular value decomposition for polynomial systems*, Proc. of ISSAC'95, ACM Press, 1995, 195–207.

[2] S. Gao, E. Kaltofen, J. P. May, Z. Yang and L. Zhi, *Approximate factorization of multivariate polynomials via differential equations*, Proc. of ISSAC'04, ACM Press, 2004, 167–174.

[3] M. Ochi, M-T. Noda and T. Sasaki, *Approximate greatest common divisor of multivariate polynomials and its application to ill-conditioned systems of algebraic equations*, J. Inform. Proces., 14 (1991), 292–300.

[4] M. Sanuki. *Computing multivariate approximate GCD based on Barnett's theorem*, Proc. of Symbolic-Numeric Computation 2009 (SNC 2009), 2009, 149–157.

[5] M. Sanuki and T. Sasaki, *Computing approximate GCDs in ill-conditioned cases*, International Workshop of Symbolic-Numeric Computation 2007 (SNC2007), ACM Press, 2007, 170-179, 25–27.

[6] T. Sasaki and S. Yamaguchi, *An analysis of cancellation error in multivariate Hensel construction with floating-point number arithmetic*, Proc. of ISSAC'98, ACM Press, 1998, 1–8.

[7] Z. Zeng and B. H. Dayton, *The approximate GCD of inexact polynomials part II: A multivariate algorithm*, Proc. of ISSAC'04, ACM Press, 2004, 320–327.

[8] L. Zhi and M-T. Noda, *Approximate GCD of Multivariate Polynomials*, Proc. of ASCM2000, World Scientific, 2000, 9–18.

An Optimized Path Planning of Manipulator with Spline Curves Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems*

Yusuke Shirato[1], Natsumi Oka[1], Akira Terui[2,*] and Masahiko Mikawa[3]

[1]*Master's Program in Mathematics, Graduate School of Science and Technology, University of Tsukuba, Tsukuba 305-8571, Japan*
[2]*Institute of Pure and Applied Sciences, University of Tsukuba, Tsukuba 305-8571, Japan*
[3]*Institute of Library, Information and Media Science, University of Tsukuba, Tsukuba 305-8550, Japan*

Abstract
This paper presents an advanced method for addressing the inverse kinematics and optimal path planning challenges in robot manipulators. The inverse kinematics problem involves determining the joint angles for a given position and orientation of the end-effector. Furthermore, the path planning problem seeks a trajectory between two points. Traditional approaches in computer algebra have utilized Gröbner basis computations to solve these problems, offering a global solution but at a high computational cost. To overcome the issue, the present authors have proposed a novel approach that employs the Comprehensive Gröbner System (CGS) and CGS-based quantifier elimination (CGS-QE) methods to efficiently solve the inverse kinematics problem and certify the existence of solutions for trajectory planning. This paper extends these methods by incorporating smooth curves via cubic spline interpolation for path planning and optimizing joint configurations using shortest path algorithms to minimize the sum of joint configurations along a trajectory. This approach significantly enhances the manipulator's ability to navigate complex paths and optimize movement sequences.

Keywords
Gröbner basis, Comprehensive Gröbner Systems, Quantifier Elimination, Robotics, Inverse kinematics problem, Path planning, Trajectory planning

1. Introduction

This paper discusses a method for solving the inverse kinematics problem and the optimal path planning problem for a robot manipulator. A manipulator is a robot with links corresponding to human arms and joints corresponding to human joints, and the tip is called the end-effector. The inverse kinematics problem for manipulators is to find the angle of each joint, given the position and orientation of the end-effector. The path planning problem is to find a path to move the end-effector between two specified positions [1].

When operating the manipulator, one needs to solve the inverse kinematics problem (or the path planning problem, respectively) for the desired end-effector position (or the series of positions, respectively).

Methods of solving inverse kinematics problems for manipulators by reducing the inverse kinematics problem to a system of polynomial equations and using the Gröbner basis has been proposed [2, 3, 4, 5, 6]. Solving the inverse kinematics problem using the Gröbner basis computation has an advantage that the global solution of the inverse kinematics problem can be obtained before the end-effector will actually be "moved" by simulation or other means. On the other hand, the Gröbner basis computation has the disadvantage of relatively high computational cost compared to local solution methods such as the Newton method. Furthermore, when solving a path planning problem using the Gröbner basis

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan
*This work was partially supported by JKA and its promotion funds from KEIRIN RACE.
*Corresponding author.
terui@math.tsukuba.ac.jp (A. Terui); mikawa@slis.tsukuba.ac.jp (M. Mikawa)
https://researchmap.jp/aterui (A. Terui); https://mikawalab.org/ (M. Mikawa)
0000-0003-0846-3643 (A. Terui); 0000-0002-2193-3198 (M. Mikawa)

computation, it is necessary to solve the inverse kinematics problem for each point on the path, which is even more computationally expensive.

The third and fourth authors have also previously proposed methods for solving inverse kinematics problems using Gröbner basis computations [7, 8, 9]. The authors' contributions in their previous work [9] are as follows. We have proposed a method for solving the inverse kinematics problem and the trajectory planning problem of a 3 Degree-Of-Freedom (DOF) manipulator using the Comprehensive Gröbner System (CGS) [10]. In the proposed method, the inverse kinematic problem has been expressed as a system of polynomial equations with the coordinates of the end-effector as parameters and the CGS is calculated in advance to reduce the cost of Gröbner basis computation for each point on the path. In addition, we have proposed a method for solving the inverse kinematics problems using the CGS-QE method [11], which is a quantifier elimination (QE) method based on CGS computations, to certify the existence of a (real) solution. Furthermore, we have proposed a method to certify the existence of a solution to the whole trajectory planning problem using the CGS-QE method, where the points on the trajectory are represented by parameters.

This paper proposes the following method as an extension of the previous work [9].

1. Extension of paths used in path planning problems: while the previous work has used straight lines, this paper uses smooth curves generated by the cubic spline interpolation passing through given points. Using a curved path allows us to plan paths that avoid obstacles, for example.
2. Optimization of the joint configuration obtained as the solution to the path planning problem: when solving the path planning problem, there can be multiple solutions to the inverse kinematics problem at each point on the path. In this case, the question is which of the solutions for adjacent points can be connected to minimize the sum of the configurations of the entire sequence of the joints. In this paper, we reduce this problem to the shortest path problem of a weighted graph and propose a method to compute the optimal sequence of joint configurations using shortest path algorithms.

2. Solving path planning problems in 3-DOF manipulators

The manipulator used in this paper is myCobot 280 [12] from Elephant Robotics, Inc. (hereafter referred to simply as "myCobot"). Although myCobot is a 6-DOF manipulator, we treat it as a 3-DOF manipulator by operating only the three main joints used to move the end-effector while the remaining joints are fixed, due to the computational costs for solving the inverse kinematic problem by using the CGS and the CGS-QE method.

2.1. Formulation of the forward and the inverse kinematics problems

Figure 1 shows the components and the coordinate systems of myCobot. The forward kinematics problem for myCobot is derived using the modified Denavit-Hatenberg convention (hereafter abbreviated as "D-H convention"), which is standard in robotics [1]. Let be the Cartesian coordinate system with the origin at the manipulator's base, and let (x, y, z) be the coordinates of the end-effector in . Let $_i$ be the coordinate system with the origin at the joint i. Note that Joint 0 represents the base and Joint 8 represents the end-effector.

Let $\theta_1, \theta_3, \theta_4$ be the configuration of the joints $1, 3, 4$, respectively, to be operated in this paper, and let $s_i = \sin\theta_i$, $c_i = \cos\theta_i$ for $i = 1, 3, 4$. In the following, for a set of polynomials $F = \{f_1, \dots, f_m\} \subset \mathbb{R}[c_1, s_1, c_3, s_3, c_4, s_4]$, The system of polynomial equations $f_1 = \cdots = f_m = 0$ is denoted by $F = 0$. In this case, the inverse kinematics problem is to find the solution $c_1, s_1, c_3, s_3, c_4, s_4$ to the system of polynomial

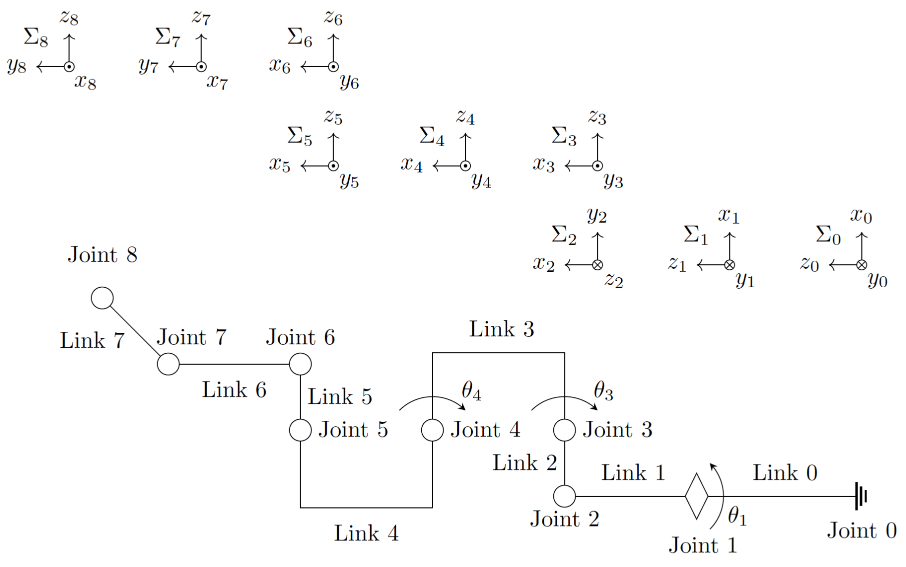


Figure 1: Components and the coordinate systems of myCobot.

equations $F = 0$ with $F = \{f_1, \dots, f_6\}$, where

$$\begin{aligned}
f_1 &= -16918s_1s_3c_4 - 16918s_1c_3s_4 - 4360s_1s_3s_4 + 4360s_1c_3c_4 - 11040s_1s_3 \\
&\quad - 6639c_1 + 100x, \\
f_2 &= 16918c_1s_3c_4 + 16918c_1c_3s_4 + 4360c_1s_3s_4 - 4360c_1c_3c_4 + 11040c_1s_3 \\
&\quad - 6639s_1 + 100y, \\
f_3 &= -16918c_3c_4 + 16918s_3s_4 - 4360c_3s_4 - 4360s_3c_4 - 11040c_3 \\
&\quad - 13156 + 100z, \\
f_4 &= s_1^2 + c_1^2 - 1, \quad f_5 = s_3^2 + c_3^2 - 1, \quad f_6 = s_4^2 + c_4^2 - 1.
\end{aligned} \tag{1}$$

Note that the system of equations $F = 0$ has parameters x, y, z in the coefficients.

With our previously proposed method [9], before solving $F = 0$, we calculate the CGS of $\langle f_1, \dots, f_6 \rangle$. Then, for a given end-effector coordinate $(x, y, z) = (\alpha, \beta, \gamma) \in \mathbb{R}^3$, determine the feasibility of such a configuration using the CGS-QE method. If the configuration is feasible, we solve $F = 0$ numerically after substituting parameters x, y, z with α, β, γ, respectively, to obtain the inverse kinematic solution.

2.2. Generating trajectories using cubic spline interpolation

In the path planning problem, a finite number of points through which the path passes are given in advance, and a trajectory is generated on the smooth path passing through these points. In our previous study, a straight line was used as the path, but in this paper, a path is generated using cubic spline interpolation [13].

Let (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) be four different points given in $\mathbb{R}^3$. For $j = 0, 1, 2$, calculate a curve C_j passing through (x_j, y_j, z_j) and $(x_{j+1}, y_{j+1}, z_{j+1})$ in the form of a function $(X_j(s), Y_j(s), Z_j(s))$ with $s \in [j, j+1]$ as a parameter. In cubic spline interpolation, $X_j(s), Y_j(s), Z_j(s)$ are cubic polynomials, respectively, satisfying the following conditions.

$$\begin{aligned}
(X_j(j), Y_j(j), Z_j(j)) &= (x_j, y_j, z_j), \quad j = 0, 1, 2, \\
(X_j(j+1), Y_j(j+1), Z_j(j+1)) &= (x_{j+1}, y_{j+1}, z_{j+1}), \quad j = 0, 1, 2, \\
(X_j'(j+1), Y_j'(j+1), Z_j'(j+1)) &= (X_{j+1}'(j+1), Y_{j+1}'(j+1), Z_{j+1}'(j+1)), \quad j = 0, 1, \\
(X_j''(j+1), Y_j''(j+1), Z_j''(j+1)) &= (X_{j+1}''(j+1), Y_{j+1}''(j+1), Z_{j+1}''(j+1)), \quad j = 0, 1.
\end{aligned} \tag{2}$$

Table 1
Test points for the inverse kinematics problem. The unit of the coordinates is [mm].

Test	(x_0, y_0, z_0)	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
1	$(-100, -100, 100)$	$(0, -150, 50)$	$(50, -100, 50)$	$(100, 0, 0)$
2	$(-150, -100, 150)$	$(0, -100, 100)$	$(100, -50, 50)$	$(100, 100, 0)$
3	$(-250, 0, 0)$	$(-150, 0, 50)$	$(-150, 100, 150)$	$(250, 100, 100)$
4	$(100, 50, 0)$	$(50, 100, 150)$	$(-150, 150, 100)$	$(-150, 50, 50)$
5	$(100, 100, -50)$	$(50, 100, 0)$	$(-100, 100, 50)$	$(-150, -50, 100)$
6	$(150, 150, 0)$	$(50, 50, 100)$	$(-50, -50, 100)$	$(-150, -150, 50)$

In this paper, in addition to the conditions in eq. (2), we find the *natural* cubic Spline interpolation that satisfies

$$X_0''(0) = X_3''(3) = Y_0''(0) = Y_3''(3) = Z_0''(0) = Z_3''(3) = 0. \tag{3}$$

2.3. Solving the inverse kinematics problem

For the trajectory C obtained in the previous section, we solve the inverse kinematics problem using the following procedure.

First, using the CGS-QE method, we determine the existence of solutions to the inverse kinematics problem for each curve C_j ($j = 0, 1, 2$) that constitutes the path to move the end-effector. In eq. (1), replace x, y, z in each equation by $X_j(s), Y_j(s), Z_j(s)$, respectively, to obtain a system of polynomial equations with s as parameter. Let $F_j'(s)$ be the result.

Next, we apply the CGS-QE method to $F_j'(s)$ to determine whether the system of equations $F_j'(s) = 0$ has real solutions within the parameter $s \in [j, j+1]$. If $F_j'(s) = 0$ has real solutions within the parameter $s \in [j, j+1]$ (i.e., within $s \in [0, 3]$ throughout for s), we calculate the trajectory of the end-effector's position for time t. Let $T = 3u$ (where u is a positive integer) and, for time $t = 0, \dots, T$, let $s = f(t)$ where f is a continuous function of $[0, T] \to [0, 3]$. To ensure that the end-effector has no velocity and acceleration at the beginning and end of the trajectory, we obtain f as a polynomial of the smallest degree possible to satisfy $f'(t) = f''(t) = 0$ at $t = 0$ and T. Then, we obtain $f(t) = 3\left(\frac{18t^5}{T^5} - \frac{45t^4}{T^4} + \frac{30t^3}{T^3}\right)$ [14].

Finally, we solve the inverse kinematics problem $t = 0, \dots, T$. For $j = 0, 1, 2$ and $t = ju, \dots, (j+1)u$, the configuration of the joints is obtained by solving the system of polynomial equations

$$F_j'(f(t)) = 0. \tag{4}$$

Assuming that the time required to solve the inverse kinematics problem (4) at each value of t is constant, then the computation time for trajectory planning is expected to be proportional to T.

2.3.1. Experimental results

We have implemented the above method and conducted experiments. The implementation of the inverse kinematics computation was performed as follows: the CGS computation was performed using an algorithm by Kapur et al. [15] with the implementation by Nabeshima [16] on the computer algebra system Risa/Asir [17]. The existence of the root of the inverse kinematics problem by the CGS-QE method was verified using Risa/Asir with accompanying Wolfram Mathematica 13.1 [18] for simplifying the expression.

We have given the set of test points as shown in Table 1, whose unit of the coordinates is [mm]. The experiments were conducted in the following environment: Intel Xeon Silver 4210 3.2 GHz, RAM 256 GB, Linux Kernel 5.4.0, Risa/Asir Version 20230315, Wolfram Mathematica 13.1.

Table 2 shows the results of the experiments for $T = 10$ and $T = 50$. The comprehensive Gröbner system for eq. (1) uses precomputed values. If any part of the generated spline for the given coordinates falls outside the feasible region of myCobot, an error is displayed. The average computation time does

Table 2
Results of path planning time in seconds.

Test	$T = 10$ [s]	$T = 50$ [s]
1	1.293	7.836
2	1.342	6.940
3	1.296	8.049
4	1.574	7.616
5	1.265	7.155
6	Fail	Fail
Average	1.354	7.429

not consider the computation time of tests that resulted in an error. The experimental results show that the computation time is considered to be approximately proportional to T. In Test 6, it appears that part of the path exceeds the feasible region, causing the computation to terminate with an error.

2.4. Solving optimal path planning problem

The system of equations (4) may have several different solutions at each time t. For time $t = 0, \dots, T$, suppose that there exist $k_{j,t}$ solutions to the configuration θ_j of joint j ($j = 1, 3, 4$) at t such that $\theta_{j,t}^{(1)}, \dots, \theta_{j,t}^{(k_{j,t})}$. The sum of the configuration changes of the joints from time $t = 1$ to T differs depending on the choice of the solution from $\theta_{j,t}^{(1)}, \dots, \theta_{j,t}^{(k_{j,t})}$ at each time t. In order to move each joint more smoothly, it is desirable to select a solution that minimizes the sum of the configuration changes of the joints on the path. For this purpose, for $j = 1, 3, 4$, we define a weighted graph $G_j = (V_j, E_j)$, where $V_j = \{\theta_{j,t}^{(k)} \mid t \in \{0, \dots, T\}, k \in \{1, \dots, k_{j,t}\}\}$ is the set of vertices corresponding to the joint configuration at each time t, and $E_j = \{(\theta_{j,t}^{(k_1)}, \theta_{j,t+1}^{(k_2)}) \mid 1 \leq k_1, k_2 \leq k_{j,t}, t \in \{0, \dots, T-1\}\}$ is the set of edges connecting the vertices at adjacent times. Then, in G_j, we reduce the problem of finding the optimal path to the one of finding the shortest path from $\theta_{j,0}^{(k)}$ to $\theta_{j,T}^{(k)}$. We have solved the shortest path problem using the following methods.

Method 1 For $t = 0, \dots, T-1$, find the sequence of joint configurations $\{\theta_{j,t}^{(k)} \mid t = 0, \dots, T\}$ satisfying the minimum distance between adjacent joint configurations such that $\min\{|\theta_{j,t}^{(k_1)} - \theta_{j,t+1}^{(k_2)}| \mid 1 \leq k_1 \leq k_{t,k_{j,t}}, 1 \leq k_2 \leq k_{t+1,k_{j,t+1}}\}$.

Method 2 In the graph G_j, find the shortest path starting at $\theta_{j,0}^{(k)}$ ($k = 1, \dots, k_{j,0}$) using the Dijkstra method [19], then find the shortest path with the minimum length among G_js.

The arithmetic complexity of each method above is estimated as follows. Let T be the number of points in the trajectory, and assume that the number of solutions to the inverse kinematics problem at each point is constant at d. The complexity of Method 1 is $O(Td)$. On the other hand, the complexity of Method 2, using a binary heap, is estimated to be $O(Td^2 \log(Td))$.

2.4.1. Experimental results

We have implemented the above method and conducted experiments. The implementation of each procedure was based on an implementation [20] in the Python programming language. The experiments were conducted in the following environment: A virtual machine with RAM 13.2 GB, Ubuntu 22.04.3 LTS, Python 3.10.2 on VMware Workstation 16 Player, on the host environment with Intel Core i7-1165G7, RAM 16 GB, Windows 11 Home.

The test points are composed of a total of $T = 15$ points, obtained by dividing each segment of the cubic spline curve passing through each point in Table 3 into 5 parts. The number of solutions to the inverse kinematics problem at each point in each test segment is $d = 4$.

Table 3
Test points for the optimal path planning problem. The unit of the coordinates is [mm].

Test	(x_0, y_0, z_0)	(x_1, y_1, z_1)	(x_2, y_2, z_2)	(x_3, y_3, z_3)
1	$(-100, -100, 100)$	$(0, -150, 50)$	$(50, -100, 50)$	$(100, 0, 0)$
2	$(-150, -100, 150)$	$(0, -100, 100)$	$(100, -50, 50)$	$(100, 100, 0)$
3	$(-250, 0, 0)$	$(-150, 0, 50)$	$(-150, 100, 150)$	$(250, 100, 100)$
4	$(100, 50, 0)$	$(50, 100, 150)$	$(-150, 150, 100)$	$(-150, 50, 50)$
5	$(100, 100, -50)$	$(50, 100, 0)$	$(-100, 100, 50)$	$(-150, -50, 100)$

Table 4
The sum of the joint variations [rad].

Test	Method 1 [rad]	Method 2 [rad]
1	8.3142	4.4013
2	11.5985	9.8986
3	15.0005	13.6731
4	8.5828	6.3277
5	7.1897	5.7108
Average	10.1371	8.0023

Table 5
Computing time of the optimal path computation [10^{-4}s].

Test	Method 1 [10^{-4}s]	Method 2 [10^{-4}s]
1	1.96	39.4
2	2.16	42.8
3	2.90	33.0
4	2.90	40.1
5	4.44	36.4
Average	2.87	31.3

Table 4 shows the sum of the joint variations [rad]. From the average values of each result, we see that Method 2 (with the Dijkstra method) reduces the total amount of joint rotation.

Table 5 shows the computation time for each method. From the average computation times, it can be seen that Method 1 is approximately 10 times faster than Method 2. Compared to the overall computation time for route planning shown in Table 2, the computation time for optimal route selection is considered sufficiently short, even for using Method 2 (Dijkstra method).

3. Concluding remarks

We have proposed a method for trajectory planning of robot manipulators using computer algebra, where the path is provided by cubic spline interpolation. Furthermore, we have proposed a method to optimize the joint configuration of the manipulator by solving the shortest path problem in a weighted graph. The experimental results have shown that the proposed methods can be used to plan the trajectory of the manipulator and optimize the joint configuration.

Future work includes the following.

1. Regarding path planning using cubic splines, it has been pointed out that some parts of the generated path may deviate from the feasible region. In the future, it is necessary to consider methods that ensure the generated trajectory stays within the feasible region while meeting given constraints, such as avoiding obstacles. In response to this, the authors are currently proposing a path-planning method that uses Bézier curves to generate trajectories that remain within the feasible region [21].

2. Instead of representing the trajectory on the path, it is desired to express any point on the path using parameters and ensure the solution to the inverse kinematics problem for that point. For linear paths, a method has already been proposed by the authors [9], and this will be extended to curves given by cubic splines.
3. Regarding optimal path planning using shortest path calculations on graphs, we have used an implementation of Dijkstra's algorithm using a binary heap in this paper. However, one method to further improve computational efficiency is to use Dijkstra's algorithm with a Fibonacci heap [22]. For example, using an implementation of Dijkstra's algorithm with a Fibonacci heap can be considered to improve computational efficiency.
4. To extend the proposed method to a 6-DOF manipulator. Although myCobot is operated with 3 Degree-Of-Freedom in this paper, it originally had 6 Degree-Of-Freedom, so it is desirable to implement methods for motion planning and path planning with 6 Degree-Of-Freedom.

References

[1] B. Siciliano, O. Khatib, Springer Handbook of Robotics, 2nd ed., Springer, 2016. doi:10.1007/978-3-319-32552-1.

[2] J.-C. Faugère, J.-P. Merlet, F. Rouillier, On solving the direct kinematics problem for parallel robots, Research Report RR-5923, INRIA, 2006. URL: https://hal.inria.fr/inria-00072366.

[3] C. M. Kalker-Kalkman, An implementation of Buchbergers' algorithm with applications to robotics, Mech. Mach. Theory 28 (1993) 523–537. doi:10.1016/0094-114X(93)90033-R.

[4] S. Ricardo Xavier da Silva, L. Schnitman, V. Cesca Filho, A Solution of the Inverse Kinematics Problem for a 7-Degrees-of-Freedom Serial Redundant Manipulator Using Gröbner Bases Theory, Mathematical Problems in Engineering 2021 (2021) 6680687. doi:10.1155/2021/6680687.

[5] T. Uchida, J. McPhee, Triangularizing kinematic constraint equations using Gröbner bases for real-time dynamic simulation, Multibody System Dynamics 25 (2011) 335–356. doi:10.1007/s11044-010-9241-8.

[6] T. Uchida, J. McPhee, Using Gröbner bases to generate efficient kinematic solutions for the dynamic simulation of multi-loop mechanisms, Mech. Mach. Theory 52 (2012) 144–157. doi:10.1016/j.mechmachtheory.2012.01.015.

[7] N. Horigome, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gröbner Bases, in: A. M. Bigatti, J. Carette, J. H. Davenport, M. Joswig, T. de Wolff (Eds.), Mathematical Software – ICMS 2020, Springer International Publishing, Cham, 2020, pp. 3–13. doi:10.1007/978-3-030-52200-1_1.

[8] S. Otaki, A. Terui, M. Mikawa, A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Real Quantifier Elimination based on Comprehensive Gröbner Systems, Preprint, 2021. doi:10.48550/arXiv.2111.00384, arXiv:2111.00384.

[9] M. Yoshizawa, A. Terui, M. Mikawa, Inverse Kinematics and Path Planning of Manipulator Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems, in: Computer Algebra in Scientific Computing. CASC 2023, volume 14139 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 393–419. doi:10.1007/978-3-031-41724-5_21.

[10] V. Weispfenning, Comprehensive Gröbner Bases, J. Symbolic Comput. 14 (1992) 1–29. doi:10.1016/0747-7171(92)90023-W.

[11] R. Fukasaku, H. Iwane, Y. Sato, Real Quantifier Elimination by Computation of Comprehensive Gröbner Systems, in: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 173–180. doi:10.1145/2755996.2756646.

[12] Elephant Robotics Co., Ltd., myCobot 280 M5, 2023. URL: https://www.elephantrobotics.com/mycobot-280-m5-2023, accessed 2024-05-04.

[13] G. Farin, Curves and Surfaces for CAGD: A Practical Guide, The Morgan Kaufmann Series in Computer Graphics, 5th ed., Morgan Kaufmann, 2002. doi:10.1016/B978-1-55860-737-8.X5000-5.

[14] K. M. Lynch, F. C. Park, Modern Robotics: Mechanics, Planning, and Control, Cambridge University Press, 2017.

[15] D. Kapur, Y. Sun, D. Wang, An efficient method for computing comprehensive Gröbner bases, J. Symbolic Comput 52 (2013) 124–142. doi:10.1016/j.jsc.2012.05.015.

[16] K. Nabeshima, CGS: a program for computing comprehensive Gröbner systems in a polynomial ring [computer software], 2018. URL: https://www.rs.tus.ac.jp/~nabeshima/softwares.html, accessed 2024-05-04.

[17] M. Noro, T. Takeshima, Risa/Asir — A Computer Algebra System, in: ISSAC '92: Papers from the International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery, New York, NY, USA, 1992, pp. 387–396. doi:10.1145/143242.143362.

[18] Wolfram Research, Inc., Mathematica, Version 13.1 [computer software], 2022. URL: https://www.wolfram.com/mathematica, accessed 2024-05-04.

[19] E. W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271. doi:10.1007/BF01386390.

[20] simonritchie, Compute the shortest path of a graph using Dijkstra method and Python (in Japanese), 2023. URL: https://qiita.com/simonritchie/items/216eae753fc393da52af, accessed: 2024-05-04.

[21] R. Hatakeyama, A. Terui, M. Mikawa, Towards Trajectory Planning of a Robot Manipulator with Computer Algebra using Bézier Curves, in: SCSS 2024 Work-in-progress Proceedings, Open Publishing Association, 2024. To appear.

[22] S.-L. Guo, J. Duan, Y. Zhu, X.-C. Li, T.-W. Chen, Improved dijkstra algorithm based on fibonacci heap for solving the shortest path problem with specified nodes, in: Computer Science and Artificial Intelligence: Proceedings of the International Conference on Computer Science and Artificial Intelligence (CSAI2016), World Scientific, 2017, pp. 52–61. doi:10.1142/9789813220294_0008.

Reasoning about the Embedded Shape of a Qualitatively Represented Curve[⋆]

Kazuko Takahashi[1]

[1]*Kwansei Gakuin University, 1 Gakuen Uegahara, Sanda, Japan*

Abstract

This paper addresses the problem of embedding curves, represented qualitatively as sequences of primitive segments, onto a two-dimensional plane. These primitive segments are directed curved segments with intrinsic direction and convexity. We define a symbolic expression to each segment, and by connecting these segments, we can derive a symbolic expression that represents an abstract shape of a smooth continuous curve. There are an infinite number of embeddings of the derived curve on a two-dimensional plane, since precise information such as coordinates are missing. However, for some shape of curves, any embedding forms a spiral, which is undesirable when the curve represents the boundary of a natural object. We propose a method for judging whether it is possible to draw a curve not in a spiral form on a two-dimensional plane by checking the segment orientation.

Keywords

qualitative spatial reasoning, curved line, embedding on a plane

1. Introduction

Qualitative Spatial Reasoning (QSR) is a method that gives a symbolic expression to a spatial object or the relationships between objects, focusing on a specific aspect of the spatial data, and that conducts reasoning on this expression [1, 2, 3]. The approach requires no big data and has less computational complexity, since it does not treat the numerical data. It also enables reasoning that suits human recognition. The focused aspects involve relative location, direction, size, distance of objects, the shapes of an object, and so on. Systems that combine more than one aspect are also proposed.

Previously, we proposed a qualitative representation that describes the outline of a curve as a sequence of segments [4]. There, we defined the connection rule of curved primitive segments to obtain a smooth continuous curve. On the other hand, there are an infinite number of embeddings of the obtained curve on a two-dimensional plane, since precise information such as coordinates are missing. If an embedding forms a spiral, it is not realistic as a boundary of an object in nature.

In this paper, we discuss whether there exists a way to embed a curve so that it does not form a spiral and show the judgment method by introducing the reduction rules on the sequence of orientations of a curve.

This paper is organized as follows. In Section 2, we describe fundamental concepts. In Section 3, we discuss the embedding of a curve in a qualitative representation on a two-dimensional plane. In Section 4, we propose the method of judging whether there exists an embedding that does not form a spiral. In Section 5, we compare our work with the related works. Finally, in Section 6, we show our concluding remarks.

SCSS 2024: 10th International Symposium on Symbolic Computation in Software Science, August 28–30, 2024, Tokyo, Japan

[⋆]You can use this document as the template for preparing your publication. We recommend using the latest version of the ceurart style.

[*]Corresponding author.

ktaka@kwansei.ac.jp (K. Takahashi)
https://ist.ksc.kwansei.ac.jp/~ktaka/LABO/ (K. Takahashi)
0000-0002-5572-7747 (K. Takahashi)

2. Fundamental Concepts

Let $CURVES$ be a set of directed curved segments with a unique direction and curvature on a two-dimensional plane. For $\boldsymbol{X} \in CURVES$, we represent the qualitative shape of $\boldsymbol{X}$ focusing on its intrinsic direction and convexity, ignoring the precise size and the exact curvature.

Let $S_v = \{n, s\}$, $S_h = \{e, w\}$, $Conv = \{cx, cc\}$ and $Dir = S_v \cup S_h$. The symbols n, s, e and w indicate the north, south, east and west directions, respectively, and cx and cc indicate convex and concave, respectively. The direction exactly in the middle between north and south (east and west) is regarded as either n or s (e or w, resp.). Straight lines are not considered. For $\boldsymbol{X} \in CURVES$, $X = (V, H, C)$ is said to be *the qualitative representation of* $\boldsymbol{X}$ where $V \in S_v, H \in S_h$ and $C \in Conv$. V, H and C show the vertical direction, horizontal direction and the convexity of $\boldsymbol{X}$. For $\boldsymbol{X}, \boldsymbol{Y} \in CURVES$, let $X = (V, H, C)$ and $Y = (V', H', C')$ be qualitative representations of $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively. We define the relation $\sim$ on $CURVES$ as follows: $\boldsymbol{X} \sim \boldsymbol{Y}$ iff $V = V'$, $H = H'$ and $C = C'$. Then $\sim$ is an equivalence relation on $CURVES$. As a result, $CURVES$ is classified into eight equivalence classes which are jointly exhaustive and pairwise disjoint. We denote the set of these equivalence classes as $\mathcal{S}$, that is, $\mathcal{S} = CURVES/\sim$. Then, $\boldsymbol{X} \in CURVES$ is mapped to $X \in \mathcal{S}$.

In this paper, a smooth continuous curve without a self-intersection is called an *scurve*. We connect multiple segments in $\mathcal{S}$ to create an scurve.

For $X \in \mathcal{S}$, its initial and terminal points are indicated by $init(X)$ and $term(X)$, respectively. For $X, Y \in \mathcal{S}$, if an scurve is obtained by considering that $init(Y)$ and $term(X)$ are identical, then X and Y are said to be *directly connectable*, and the outcome of the connection is represented as $X \cdot Y$. For $X, Y \in \mathcal{S}$, if $X = Y$, then they are directly connectable and the result is regarded as a single segment without a cusp, since the precise curvatures of X and Y are ignored. When X and Y are directly connectable, and $X \neq Y$, the connection of X and Y create inflection or extremum points via direct connections.

For $X_1, \ldots, X_n \in \mathcal{S}$ ($n \geq 2$), if for each j such that $1 \leq j \leq n-1$, X_j and X_{j+1} are directly connectable, then we obtain an scurve by directly connecting X_j and X_{j+1}, and the outcome of the connections is represented as $X_1 \cdot \ldots \cdot X_n$. As a result, scurve is a sequence of qualitative representations of curved segments.

For example, $X = (n, e, cx)$ and $Y = (n, e, cc)$ are directly connectable (Figure 1(a)), and $X = (n, e, cx)$ and $Y = (s, e, cx)$ are directly connectable (Figure 1(b)). On the other hand, $X = (n, e, cx)$ and $Y = (s, e, cc)$ are not, since a cusp is created at their connection (Figure 1(c)); but if we insert $Z = (s, e, cx)$ between X and Y, then we get an scurve $X \cdot Z \cdot Y$ (Figure 1(d)).

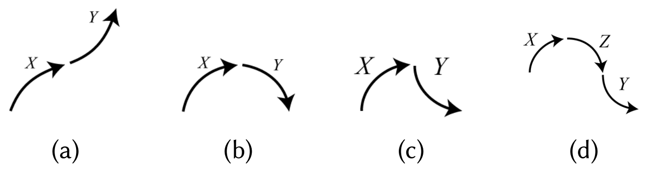


Figure 1: Connection of segments.

3. Embedding on a Two-Dimensional Plane

In the following, 'embedding of X' means an assignment of one $\boldsymbol{X} \in CURVES$ to $X \in \mathcal{S}$. It is defined as follows:

1. Let $\boldsymbol{X} \in CURVES$ be a curved segment on a two-dimensional plane of which $X \in \mathcal{S}$ is its qualitative representation. (Note that there are infinite number of $\boldsymbol{X}$'s.) Then $\boldsymbol{X}$ is said to be *an embedding of* X. $init(\boldsymbol{X})$ and $term(\boldsymbol{X})$ represent the locations of the initial point and the terminal point of $\boldsymbol{X}$ on a two-dimensional plane, respectively.

2. Let $X_1 \cdot \ldots \cdot X_n$ be an scurve $X_1 \cdot \ldots \cdot X_n$, and $\boldsymbol{X}_i$ $(1 \leq i \leq n)$ be *an embedding of* X_i. For all i such that $1 \leq i \leq n-1$, if $term(\boldsymbol{X}_i)$ and $init(\boldsymbol{X}_{i+1})$ are located in the same position, then $\boldsymbol{X}_1 \cdot \ldots \cdot \boldsymbol{X}_n$ is said to be *an embedding of an scurve* $X_1 \cdot \ldots \cdot X_n$.

For example, Figure 2 shows two kinds of embeddings of $X \cdot Y$ where $X = (n, e, cx)$ and $Y = (s, e, cx)$. The relative directions of the locations of $term(\boldsymbol{Y})$ regarding $init(\boldsymbol{X})$ are (n, e) and (s, e) in Figure 2(a) and Figure 2(b), respectively.

If an embedding of an scurve forms a spiral, it is not desirable, when an scurve corresponds to a boundary of an actual object. However, there exists an scurve which cannot be drawn in a non-spiral form, no matter how it is drawn. Here, we introduce a concept of an orientation of an scurve on a symbolic expression, and discuss the shape of its embedding by checking the orientation. We show how to determine whether there exists an embedding that does not form a spiral on a two-dimension plane, for a given scurve. For this purpose, we introduce a concept of open/closed embedding.

For $X, Y \in \mathcal{S}$, let C be an embedding of an scurve from X to Y on a two-dimensional plane, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ be embeddings of X and Y, respectively. And C' be an infinite-length curve that is obtained by extending C in both directions in a manner such that the curvature of $\boldsymbol{X}$ at $init(\boldsymbol{X})$ and that of $\boldsymbol{Y}$ at $term(\boldsymbol{Y})$ are preserved. If C' has a self-intersection point, then the embedding is said to be *closed*; otherwise, it is *open*. Figure 3 shows open (a) and closed (b) embeddings of an scurve $X \cdot Z \cdot Y$ where $X = (n, e, cx)$, $Z = (s, e, cx)$ and $Y = (s, w, cc)$.

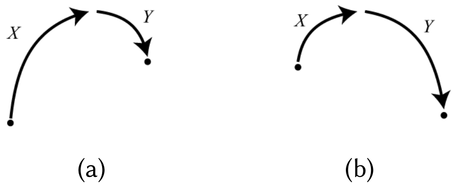


Figure 2: Different embeddings of $X \cdot Y$.

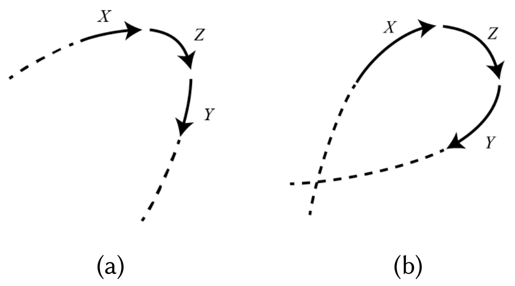


Figure 3: Open/closed embedding of $X \cdot Z \cdot Y$.

If there is an open embedding of an scurve, then the scurve is said to be *admissible*. The empty sequence ϵ is considered to be admissible.

4. Admissibility

4.1. Reduction

For $X \in \mathcal{S}$, its orientation is defined either as clockwise (+) or anti-clockwise (−). Moreover, the orientation of an scurve is defined as a sequence of orientations of the segments that configure it.

- For $X \in \mathcal{S}$,
  $orn(X) = ' + '$ iff $X = (n, e, cx)$, (s, e, cx), (s, w, cc) or (n, w, cc); $orn(X) = ' - '$ iff $X = (s, w, cx)$, (s, e, cc), (n, e, cc) or (n, w, cx).
- For $X_1, \ldots, X_n \in \mathcal{S}$,
  $orn(X_1 \cdot \ldots \cdot X_n) = orn(X_1) \ldots orn(X_n)$.

We define the function inv on the set $\{+, -\}$ that assigns the opposite orientation: $inv(+) = -$ and $inv(-) = +$.

For an scurve p, the difference of the numbers of $+$ and $-$ that appear in $orn(p)$ is said to be *rotation difference of* p.

Some specific subsequences in the orientation of an scurve do not affect the judgment of its admissibility. We consider a shorter sequence by removing these parts. There are two reduction rules: the

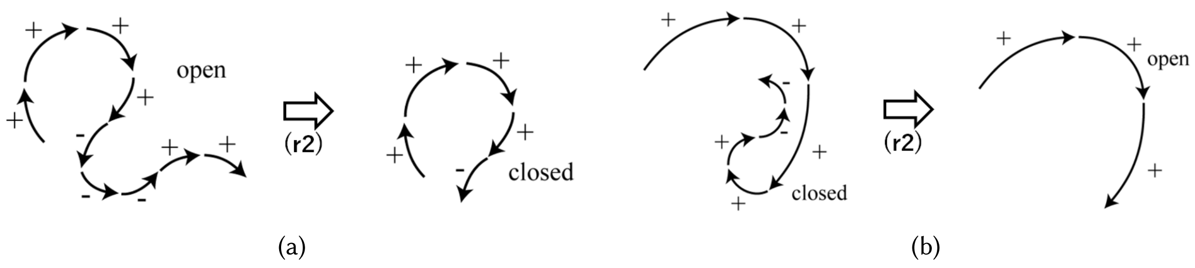


Figure 4: Examples in which admissibility is not preserved.

subsequence $+-+$ (or $-+-$) is reduced to $+$ (or $-$, resp.), and the subsequence $++--$ (or $--++$) is reduced to the empty sequence ϵ.

[Reduction rules]

Let σ_1 and σ_2 be sequences of orientations and $s_1, s_2, s_3, s_4 \in \{+, -\}$.

(r1) If $s_1 = s_3 = inv(s_2)$, then $\sigma_1 s_1 s_2 s_3 \sigma_2$ is reduced to $\sigma_1 s_1 \sigma_2$.

(r2) If $s_1 = s_2 = inv(s_3) = inv(s_4)$ and ($\sigma_1, \sigma_2 = \epsilon$ or $\sigma_1, \sigma_2 \neq \epsilon$), then $\sigma_1 s_1 s_2 s_3 s_4 \sigma_2$ is reduced to $\sigma_1 \sigma_2$.

For a sequence of orientations σ, a sequence of orientations obtained by applying the reduction rules as far as possible is said to be *a reduced form of* σ.

Note that the condition on σ_1 and σ_2 in (r2) are necessary. It means that if only one of σ_1 and σ_2 is an empty sequence, then admissibility of p is not always the same with that of p'. We show two examples that illustrate this situation.

1. Assume that $p = X_1 \cdot \ldots \cdot X_9$ where $orn(p) = + + + + - - - + +$. If we reduce the part $- - ++$ to obtain $p' = X_1 \cdot \ldots \cdot X_5$, then $orn(p') = + + + + -$. In this case, p is admissible whereas p' is not (Figure 4(a)).
2. Assume that $p = X_1 \cdot \ldots \cdot X_7$ where $orn(p) = + + + + + - -$. If we reduce the part $+ + --$ to obtain $p' = X_1 \cdot X_2 \cdot X_3$, then $orn(p') = + + +$. In this case, p is not admissible whereas p' is (Figure 4(b)).

For any sequence of orientations σ and any of its reduced form, the following properties hold, which are easily proved.

Proposition 1. 1. *(termination)*
The reduction procedure terminates.
2. *(rotation difference preservation)*
The rotation difference is preserved in the reduction.
3. *(reduced form)*
Let s be either $+$ or $-$. A reduced form is $\sigma_1 \sigma_2 \sigma_3$ where σ_2 is a nonempty sequence of s, and σ_1 and σ_3 are the sequences of at most two $inv(s)$.

Generally, *a reduced form of* σ is not unique. For example, when $\sigma = + + + - - + --$, if we apply (r2) first, then we get the reduced form ϵ; whereas if we apply (r1) first, then we get the reduced form $+-$. However, admissibility of these reduced forms are the same.

In addition to Proposition 1, reduction has a significant property of preserving admissibility.

Theorem 2 (admissibility preservation). *An scurve is admissible if and only if its reduced form is admissible.*

4.2. Judgment of Admissibility

For a given scurve p, we determine its admissibility by checking its orientation.

Let $p = X_1 \cdot \ldots \cdot X_n$ be a reduced form of a given scurve, and k be its rotation difference.

Generally it is known that if the rotation angle of a curve is greater than or equal to 2π, then it forms a spiral and may have a self-intersection point on a two-dimensional plane. If $k \geq 4$, the rotation angle of an scurve is greater than or equal to 2π; in this case, p is not admissible. Therefore, it is enough to investigate the admissibility in the cases for $k \leq 3$.

When n is more than eleven, $k \geq 4$ always holds, since there exist at most two segments at each end of a sequence that have the opposite orientation to those in the center of the sequence. It follows that any embedding of p is closed, and thus p is not admissible. When n is less than twelve, we investigate the admissibility of all possible orientations for scurves [5, 6]. Due to the symmetry of the orientations $+$ and $-$, and that of the order of the sequence, symmetric orientations need not be investigated. The introduction of the reduction significantly decreases the number of sequences to be checked, since the length is shortened and the reduced forms are restricted as is shown in Proposition 1. For example, we should examine four cases when n is six, and only one case when it is eleven. And we conclude that for any scurve, its admissibility can be determined by the sequence of its orientation, and we have gotten the following theorem.

Theorem 3. *p is admissible if and only if $k \leq 3$, where k is the rotation difference of the orientation of p.*

5. Related Works

Embedding of curves and their intersection are frequently handled in geometry or graph theory. In geometry, shapes with strict curvatures are considered; and in graph theory, connectivity between nodes is the main target to be discussed and convexity of an edge is out of focus. The QSR approach taken in this paper treats these issues from yet another viewpoint; it is suitable for human's recognition of abstract shapes and reasoning on an abstract level.

Although there have been lots of research on QSR, few of them focus on shapes, especially on curves. Several systems in these works divide the boundary of an object into segments and represent its shape by lining up the symbols corresponding to the segments [7, 8, 9]. Segments are usually equipped with information of its shape related to their subsequent segments. Additional information such as relative length and angle may be added to each segment [10, 11].

Several QSR systems have been proposed which focus on relative directions. Moratz et al. proposed OPRA that represents the relative direction of spatial entities as a ternary relation [12, 13]. In OPRA, a primitive object is considered as a vector with its initial point and terminal point, which has a similar feature with our formalization. However, the primitive object in OPRA does not have a convexity as an attribute, which means that OPRA cannot be applied to the generation of a smooth curve by connecting objects.

6. Conclusion

In this paper, we have discussed the treatment of curves in a symbolic expression, focusing on the admissibility of the curves. In conclusion, we have shown that the admissibility of a curve can be determined by its orientation sequence: if the rotation difference is less than or equal to three, the curve is admissible. We have introduced reduction rules that significantly decreases the number of sequences to be checked. This framework provides a novel approach for reasoning about the shapes of curves on a two-dimensional plane, ensuring that they do not form spirals.

It is to be considered to improve the reasoning system by relaxing the conditions on the application of the reduction rules. In addition, we are considering formalization of the obtained result as a QSR system, and also verification using proof assistant systems to certify the proofs.

References

[1] A. Cohn, J. Renz, Qualitative spatial representation and reasoning, in: Handbook of Knowledge Representation, Elsevier, 2008.

[2] J. Chen, A. Cohn, D. Liu, S. Wang, J. Ouyang, Q. Yu, A survey of qualitative spatial representations, The Knowledge Engineering Review 30 (2013) 106–136.

[3] M. Sioutis, D. Wolter, Qualitative spatial and temporal reasoning: current status and future challenges, in: Proceedings of the 30th International Joint Conference on Artificial Intelligence, 2021, pp. 4594–4601.

[4] Y. Taniuchi, K. Takahashi, Spatial representation and reasoning about fold strata: A qualitative approach, in: 15th International Conference, ICAART 2023, Revised Selected Papers, 2024, pp. 244–266.

[5] K. Takahashi, Qualitative treatment of curves and judgment for their self-intersectionality (in japanese), in: IPSJ, SIG-PRO-149, 2024.

[6] K. Takahashi, Qualitative formalization of a curve on a two-dimensional plane, in: The 16th International Conference on Spatial Information Theory (COSIT 2024), to appear.

[7] M. Leyton, A process-grammar for shape, Artificial Intelligence 34 (1988) 213–247.

[8] M. Tosue, K. Takahashi, Towards a qualitative reasoning on shape change and object division, in: 14th International Conference on Spatial Information Theory (COSIT 2019), 2019, pp. 7:1–7:15.

[9] A. Galton, R. Meathrel, Qualitative outline theory, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (1999) 1061–1066.

[10] L. M. Cabedo, L. G. Abril, F. V. Morente, Z. Falomir, A pragmatic qualitative approach for juxtaposing shapes, Journal of Universal Computer Science 16 (2010) 1410–1424.

[11] Z. Falomir, A. Pich, V. Costa, Spatial reasoning about qualitative shape compositions, Annals of Mathematics and Artificial Intelligence 88 (2020) 589–621.

[12] R. Moratz, Representing relative direction as a binary relation of oriented points, in: Proceedings of the 17th Eureopean Conference on Artificial Intelligence, ECAI'2006, 2004, pp. 407–411.

[13] T. Mossakowski, R. Moratz, Qualitative reasoning about relative direction of oriented points, Artificial Intelligence 180-181 (2012) 34–45.

Acknowledgments

This research is supported by JSPS Kakenhi 24K15096.