A Survey on the Use of Machine Learning within Computer Algebra

UZMA SHAFIQ, School of Information Technology, Deakin University, Melbourne, Australia and Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, United Kingdom of Great Britain and Northern Ireland
MATTHEW ENGLAND, Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, United Kingdom of Great Britain and Northern Ireland
NAYYAR ZAIDI, School of Information Technology, Deakin University, Melbourne, Australia

This survey explores the use of Machine Learning (ML) in the field of Computer Algebra (CA), both to optimise existing CA algorithms, and to perform symbolic computation directly. Traditional symbolic methods, while mathematically rigorous, often are computationally expensive thus limiting their use in real-world applications. Recent advances have shown that data-driven techniques can address these limitations by guiding heuristic decisions, selecting optimal algorithms, predicting structural properties of algebraic objects, or even making direct symbolic computations.

We take a systematic literature review approach and uncover work in CA applications including cylindrical algebraic decomposition, Gröbner basis computation, symbolic integration, and many more. The survey compares the different ML approaches that have been employed for these tasks, ranging from decision trees to transformers. Issues uncovered by the survey include the lack of benchmark datasets for CA, which hinders the comparison of methods and the generalizability of ML models. The survey identifies the potential for explainable AI tools to help develop trust in decisions, and to drive forward CA research itself.

CCS Concepts: • **Computing methodologies → Computer algebra systems**; **Machine learning;**

Additional Key Words and Phrases: Computer algebra, symbolic computation, artificial intelligence, machine learning

ACM Reference Format:
Uzma Shafiq, Matthew England, and Nayyar Zaidi. 2026. A Survey on the Use of Machine Learning within Computer Algebra. *ACM Comput. Surv.* 58, 15, Article 391 (November 2026), 31 pages. https://doi.org/10.1145/3831699

US is supported by a joint scholarship from Coventry University and Deakin University. ME is partially supported by UKRI EPSRC project EP/T015748/1, *Pushing Back the Doubly-Exponential Wall of Cylindrical Algebraic Decomposition* (DEWCAD).
Authors' Contact Information: Uzma Shafiq (corresponding author), School of Information Technology, Deakin University, Melbourne, Victoria, Australia and Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, United Kingdom of Great Britain and Northern Ireland; e-mail: s224380786@deakin.edu.au; Matthew England, Centre for Computational Science and Mathematical Modelling, Coventry University, Coventry, United Kingdom of Great Britain and Northern Ireland; e-mail: Matthew.England@coventry.ac.uk; Nayyar Zaidi, School of Information Technology, Deakin University, Melbourne, Victoria, Australia; e-mail: nayyar.zaidi@deakin.edu.au.

ACM 0360-0300/2026/11-ART391
https://doi.org/10.1145/3831699

1 Introduction

Computer Algebra (**CA**) algorithms are deterministic procedures that are used to manipulate, simplify, and solve mathematical expressions. They have been used for carrying out many computational mathematics tasks such as symbolic integration and solving non-linear polynomial systems. Symbolic solutions are in contrast to numerical ones, which use approximations with floating point arithmetic. Numerical methods are widely used and computationally efficient. CA on the other hand often pays for its accuracy with high computation costs, limiting its use to when absolute accuracy is crucial, or when we need the greater understanding of the underlying physical system afforded by the symbolic solution.

Machine Learning (**ML**) is a subset of **Artificial Intelligence** (**AI**) that develops statistical models to draw inferences from data, allowing for systems to adapt without being provided specific instructions. ML has found application throughout science and engineering over recent years, and underpins the prominent Generative AI tools so prominent today. The reader seeking background material on classical ML is referred to Ref. [13], on deep learning to Ref. [112], and on reinforcement learning to Ref. [100].

This article presents a systematic literature review covering the past 15 years on the topic of ML applied to CA. The purpose of this survey is to support readers by highlighting key developments, current challenges, and future research directions.

1.1 ML to Generate CA

There is a body of work in the literature seeking to use ML, specifically generative AI models, to generate the results of mathematical computation directly. This work started in 2020 with the notable article of Lample and Charton [76] who trained transformers to integrate expressions and find analytical solutions to differential equations. More recently, Kera et al. have trained a transformer model to produce Gröbner bases of 0-dimensional ideals [70]. Even in this view CA is not replaced by ML: both of these papers make extensive use of existing CA tools to produce the datasets used to train the ML models, in the latter case defining a new algebraic problem for future study.

This paradigm of direct computation is not easily integrated into a workflow in place of traditional CA. Even exceptional models will not have 100% accuracy, but the guaranteed exact computation is the *unique selling point* of CA and most users would want to see this maintained. Hence, most work in the literature to date seems to have fallen into a second paradigm of ML to guide CA, as we introduce next.

1.2 ML to Guide CA

There are often choices that need to be made in CA algorithms which are not specified in the high-level algorithm description itself. For example, in **Cylindrical Algebraic Decomposition** (**CAD**), an algorithm to study non-linear polynomial systems, a variable ordering must be specified. This variable ordering greatly affects the computational cost of the algorithm, but is often irrelevant for the analysis being undertaken. Another example comes from Buchberger's algorithm to compute a Gröbner basis, a key tool to study a set of non-linear polynomial equations. The algorithm processes a queue of data: the order in which the data is processed does not affect the correctness of the end result but can lead to computational paths of wildly different lengths. Both of these algorithms are doubly exponential in the worst-case scenario, but could how we make these choices improve the average case complexity? Such choices in CA are commonly taken by either the user themselves, or a simple human-made heuristic destined by CA experts but often not subjected to scientific rigour or even scientific reporting.

The majority of this review focuses on the use of ML for optimising CA algorithms by making choices that were previously based on human expert knowledge or human-made heuristics, since

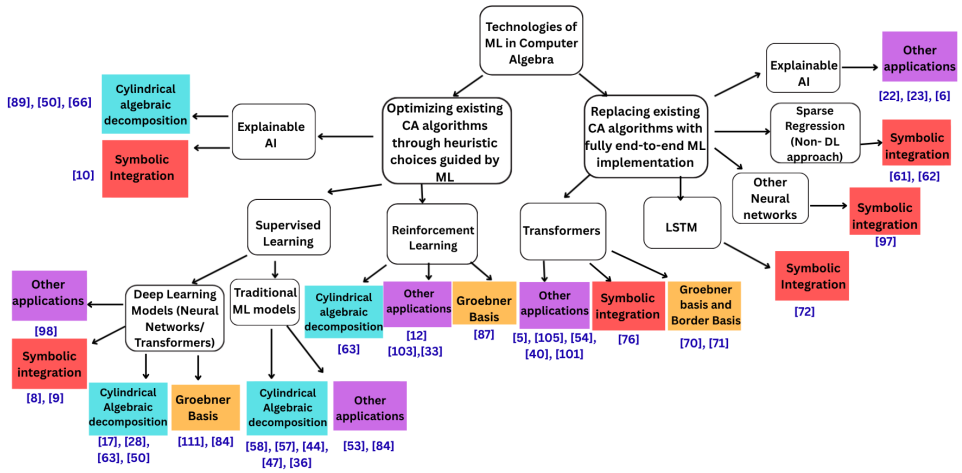

Fig. 1. Taxonomy for ML within CA employed in this article.

we find such studies more prevalent in the literature. However, we will also cover those studies available that aim at replacing CA algorithms entirely with ML-based approaches. We find examples for a range of CA algorithms in the literature. We also see a wide variety of ML tools in use: from supervised learning models trained to make distinct choices for reinforcement learning agents that adapt strategies dynamically.

1.3 Motivation

The main motivation for this survey is that, while individual studies have demonstrated the efficacy of ML in specific CA applications, there is currently no unified resource that categorises and compares this work. To the best of our knowledge, there is no similar survey in the literature. The closest examples may be the position papers of the second author [41], [42]; but these were short works, not systematic in production and containing just hand-picked examples. As ML tools grow in complexity and capability, it becomes important to critically evaluate their applicability and impact within computational algebra, and to outline future opportunities and challenges.

Furthermore, we have also introduced a taxonomy under different categories based on the application areas of CA, to understand the distinct approaches taken, although we will see some common issues emerge which we summarise at the end. This survey article also highlights the limitations and identifies research gaps for future work.

The article continues in Section 2 where we outline the methodology employed for conducting the literature review. Section 3 presents the existing research on ML cylindrical algebraic decomposition, while Section 4 offers a detailed overview of ML for Gröbner Bases. Section 5 explores additional applications of ML in other applications of CA. Finally, Section 6 summarises the key findings and discusses the limitations and research gaps in applying ML to CA.

2 Literature Review Methodology

We undertake a systematic review of the literature over the past 15 years (2010 – 2025). The motivation to select this timeline was that the earliest work applying ML to CA was 2014 [58],

Table 1. Keywords and Databases Used for Searching Relevant Literature

Category	Keywords/Databases
Keywords for ML	Machine Learning, ML, Artificial Intelligence, AI, Reinforcement Learning, RL, Deep Learning, DL.
Keywords for CA	Computer Algebra, Symbolic Computation, Non-linear Polynomials, Non-linear Systems, Non-linear Arithmetic, Gröbner basis, Groebner Basis, Grobner Basis, Buchberger's Algorithm.
Databases Searched	Scopus, IEEE Xplore, ACM Digital Library, Science Direct.
Additional Sources	arXiv, MATH AI Workshop Proceedings (NeurIPS).

Table 2. Number of Papers by Database and Keyword Combination

ML Keywords	Symbolic Computation and Variants	Non-linear Polynomials and Variants	Gröbner Basis and Variants
Scopus			
AI variants	347	222	89
ML variants	104	192	20
RL variants	9	80	1
DL variants	27	76	3
IEEE Xplore			
AI variants	65	37	1
ML variants	63	78	0
RL variants	3	3	0
DL variants	8	34	0
ACM Digital Library			
AI variants	43	8	2
ML variants	30	6	3
RL variants	6	5	3
DL variants	3	2	0
Science Direct			
AI variants	3	21	2
ML variants	6	20	2
RL variants	1	6	0
DL variants	2	10	0

so this range would capture that, and any others in years prior that we were unaware of. The review focuses on symbolic computation as the application domain only, and thus the use of CA to understand or improve ML technology is out of scope.

For the systematic search review keywords were divided into two categories. The first contained keywords to indicate the use of ML technology, and the second the application in CA (see Table 1). We seek to capture applications throughout the field of symbolic computation, but we pay particular interest to non-linear polynomial systems where there is particular scope for efficiency improvements and so include many dedicated search terms here to ensure full coverage.

The databases that were searched were Scopus, IEEE Xplore, the ACM Digital Library and Science Direct. An additional separate search was carried out on arXiv to capture work in ML conferences and workshops which are not always formally published. We also included a search on the online proceedings of MATH-AI workshop help over the last few years with NeurIPS and the AI for Math workshop at ICML. The initial results of the search are shown in Table 2.

Of the total of 1,646 entries discovered, there were 585 duplicates. The studies were then filtered based on their titles and then their abstracts. Given the relatively limited number of studies identified in this area, a single-stage screening process was considered sufficient for the purposes of this

survey. The results obtained were further filtered on the following criteria and a total of 76 studies were included:

— We searched for AI as this can be used instead of ML. However, we then screen out other forms of AI (e.g., theorem proving and SAT solvers may be considered AI but are not forms of ML).
— Studies that concern the development of neuro-symbolic ML methods are not considered within scope: neuro-symbolic ML integrate symbolic reasoning into ML systems, the complementary direction to the ML for CA focus of this review.

The filtered studies can be classified in two ways: (i) based on the ML techniques that are employed, or (ii) by the CA problems that are addressed. While Figure 1 presents a taxonomy organised by ML techniques, the survey follows the second approach. We found that organising the discussion by ML methods would result in excessive fragmentation and dense information, as many studies use overlapping or hybrid techniques. Structuring by CA applications provides a better understanding and clearer picture of how ML contributes to specific algorithmic improvements. The first taxonomy demonstrated in Figure 1 is included to give readers an alternative perspective and to highlight the breadth of ML methods explored in this domain. However organising the studies according to this taxonomy in the survey would concentrate a lot of information in certain sections which could make it more difficult for readers to follow and appreciate the progress achieved across different CA applications.

As we reviewed the remaining papers we found work focused in three main areas of CA: cylindrical algebraic decomposition, symbolic integration and Gröbner basis computation. We resent the work grouped in this way first, before a final section with the other applications identified in the literature.

3 Cylindrical Algebraic Decomposition

CAD is a tool widely used for working with non-linear polynomial systems (both equations and inequalities) defined over the real numbers. It was originally developed by Collins to solve the real quantifier elimination problem [30], where we are given a logical formula in real polynomial constraints that is quantified ($\forall, \exists$) and seek an equivalent formula without quantifiers. E.g.,

$$\exists x,\ \left(a \neq 0\ \wedge\ ax^2 + bx + c = 0\right) \equiv a \neq 0\ \wedge\ b^2 - 4ac \geq 0.$$

A CAD is produced relative to input polynomial constraints so that each has a constant truth value within each of the decomposition cells, allowing an analysis of the system by analysing one sample point per cell. CAD decomposes the real space system into cylindrical cells: a cell structure that makes it easy to test inclusion, to project and to invert cells—facilities that we can combine to answer quantifier elimination problems. For example, we solve the problem above by building a CAD over (x, a, b, c) space, testing the truth of the formula in brackets at one sample point in each, project the true cells onto (a, b, c) space, and combine their formulae to uncover the quantifier free formula on the right.

CAD is a computationally intensive algorithm, doubly exponential in the number of variables in the worst-case scenario [34]. This complexity limits the use of CAD for large problems. There are many factors that have been studied to improve the practical performance of CAD since its inception (the textbook [21] captured the first 20 years). In the last decade a key development has been the integration of CAD within logic-based AI frameworks, specifically SMT-solvers for non-linear real arithmetic [11], and the systems that use them. This led to reformulations of the CAD theory, both to better fit, and taking inspiration from, the SMT context [67], [15], [2], [85].

The CAD algorithm works by identifying additional polynomials over whose roots the truth of the input polynomials may change, and then building the decomposition according to these.

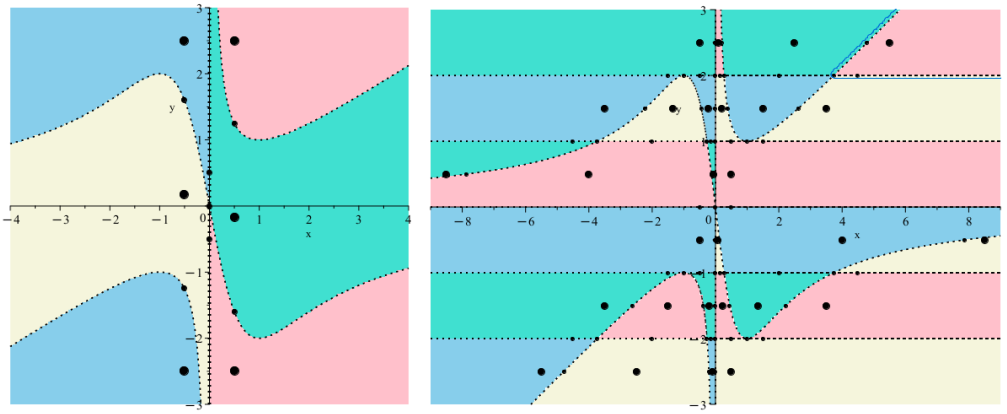

Fig. 2. CADs for $\{x^3y + 4x^2 + xy, -x^2 + 2xy - 1\}$. The left uses ordering $x \prec y$ and constructs 13 cells; the right uses $y \prec x$ and constructs 89 cells. The shaded regions show different 2D cells; and the different colour schemes the different cylinders. Sample points of each cell are marked with black discs: larger ones for the 2D cells and smaller for the lower dimensional cells.

Both the identification of the polynomials, and the building of the cells proceeds iteratively one dimension at a time. There hence has to be a determined ordering of the variables for the algorithm to run. To use the CAD for quantifier elimination we must order the variables as they are quantified, but this rarely determines the full ordering. For example, in the example above we must order x first but can have any ordering between $a, b, and c$.

3.1 Heuristic Decisions within CAD

As mentioned in the introduction, the variable ordering is one of the critical factors that affects the computational runtime of the CAD algorithm. There does not exist any exact method to ensure an optimal choice (short of trying all possibilities) and so in practice a heuristic choice is made if the user does not specify an ordering. Poor selection of the variable ordering can lead to a higher number of projection polynomials, explosive degree growth, and increased number of cells in the decomposition, and vastly different runtimes [39], [16]. This is illustrated for a simple 2d example in Figure 2 where one variable ordering results in 13 cells and the other 89, seven times more.

Due to the lack of an efficient and accurate tool, the choice of variable ordering is usually made by simple human-designed heuristics that use easily computed statistics of the input, such as *Brown's heuristic* [14] or *gmods* [37]. However, none of these heuristics may be concluded strictly superior to the others — all make mistakes for substantial subsets of problems [58]. Due to the inadequacy of existing heuristics and the dramatic effect of the choice on the following computation, CAD variable ordering has become a key case study in ML for CA literature.

Another decision that has been studied in the literature is the preconditioning of the polynomial equalities present in the system using Gröbner bases as first proposed in Ref. [20]. A Gröbner basis may be considered a simplification of the set of polynomials, and so it is perhaps not surprising that this usually benefits the subsequent CAD computation. But that is not universally the case, and identifying the value in this preconditioning on a case-by-case basis can be beneficial.

Finally, some of the more advanced CAD algorithms act iteratively by polynomial, as well as by variable [27], [15], [85]. This introduces another heuristic choice, the order in which to process polynomials, which has received far less study than the variable ordering but can also have a significant effect [43].

3.2 ML for CAD Literature

A summary of all the relevant literature identified is given in Table 3. We discuss this thematically in the following sections.

3.2.1 First Systematic ML Attempt. The first attempt at using ML to select CAD variable ordering was made in 2014 by Huang et al. [58]. They used the NLSAT dataset from [67]: a set of satisfiability problems whose atoms were non-linear polynomial constraints. The authors of Ref. [58] used only those problems involving three variables (7001 problems) and trained a simple ML model to choose which from three human-designed heuristics to follow for a given problem instance, seeking to minimise the cell counts in the resulting CADs. They reported that the ML choice was superior to following any one heuristic. Although the study opened a new paradigm for incorporating ML into CA, the proposed methodology was limited to the three variable case, and to the performance of the existing human-designed heuristics.

3.2.2 Alternative Formulations of the ML Problem. England and Florescu omitted the human-designed heuristics and had ML directly choose the variable ordering in Ref. [44]. They again used the NLSAT dataset restricted to three variable problems, meaning the ML had a choice from 6 possible orderings. This time ML models were trained to minimise CAD computation time: the ML tools outperformed the human designed heuristics.

Both [58] and [44] formulated the ML problem as classification (picking from a discrete set of options: human-designed heuristics in the former and variable orderings in the latter). One weakness of this approach is that the ML cannot take into consideration the wider ranking of orderings, i.e., even though the second-best ordering is not the correct classification it is preferable to the worst ordering. The issue is particularly prevalent since the CAD run-times tended to gather in clusters (e.g., a cluster of orderings with similar quick run-times and another with similar slow run-times). In this context, picking the right cluster is key. This problem was outlined in 2019 by Florescu and England [48] who took a first step to address it by allowing the hyper-parameter selection in their workflow to be made with respect to actual runtime, rather than classification accuracy.

In 2024 Del Río and England took the next logical step and reframed the ML problem as regression to predict the computation time with a given ordering [38]. This in turn solves the classification problem (choose the ordering we predict will be quickest). In general ML regression is more difficult than classification, but here the key evaluation metric is not regression accuracy but the classification performance of the choices that regression led to. The results demonstrated that regression could give better overall performance, but only when ML models best suited for this problem formation are used.

3.2.3 Fixing the Number of Variables. A key weakness in the early work on this problem was the training of models for problems of a fixed number of variables, usually three (although in Ref. [48] the authors increased to train a four variable model). The number of different variable orderings is factorial of the number of variables, and so the decision becomes super-exponentially harder. Although, since CAD itself is restricted by exponential complexity, there is a limit to how far we will be called upon to make such decisions.

This shortcoming was first considered in 2020 by Chen et al. [28]. The authors here proposed an iterative greedy method for generating candidate variable orderings and then an ML approach for selecting amongst these, allowing the ML to avoid the explosion in possible orderings.

Also in 2020, Ref. [17] confronted a similar issue of unbounded input size when considering the problem of choosing the order of polynomials for the adapted CAD algorithm [15]. The authors tackled this via forming an ordering of polynomials by repeated binary ML classification on pairs.

3.2.4 SMT-LIB Dataset Issues. The most commonly used dataset in the work above comes from the NRA and QF_NRA categories of the SMT-LIB (https://smt-lib.org/). This was seeded by the NLSAT dataset [67] and has been enlarged in the years since. These are problems primarily designed for solution by SMT-solvers and so lack problems with a wider range of quantifier structures that CAD can still study (for which there does not appear to exist any substantial dataset). Nevertheless, the examples are mainly sourced from real-world applications, making them valuable for benchmarking purposes. The library contains examples from fields as diverse as biology and economics, but the most prevalent examples are those created automatically by other software which calls an SMT-solver as a sub-tool, such as automated theorem provers and term-rewrite systems.

However, all the work prior to 2023 described above, all overlooked biases present in this dataset: in particular for the three-variable subset that one particular ordering is far more often the optimal one than the other five. This raises the question as to whether the ML models above are learning only this bias. The issue was simultaneously reported in two 2023 papers [56] [36] who both proposed a process of data augmentation to address the imbalance by permuting variable names. This creates new data instances, without any further cost in CA computation to label them, solving the imbalance problem and greatly increasing the dataset size. The papers described how ML models trained on a biased dataset do indeed perform poorly when testing on an unbiased one; but when trained on an unbiased augmented dataset then testing performance remains strong. So the earlier models were learning the bias and removing that unfair advantage drops performance; but performance can then be boosted back by using the larger dataset gained through data augmentation.

The problems with the dataset go beyond this particular bias however. It is described in Ref. [38] that the dataset contains many problem instances which are very similar, although not identical, perhaps differing by one coefficient in a polynomial say. This is caused for example by a theorem prover generating incremental SMT calls while developing a proof [3]. In many (but not all) cases such a small difference has no material impact on the resulting CAD calculation, which raises again the question of data leakage between training and testing. The authors of [38] address this by considering for each 3-variable problem problem the associated vector of six CAD cells counts (one for each variable ordering): where there are problems with identical such vectors all but one is discarded. This reduced the dataset to a sixth of its original size, but because these extra problems did not add information, this did not affect the learning performance.

3.2.5 Synthetic Data. The alternative to using a dataset of real-world problems is to create synthetic data. This was first attempted for CAD in 2016, when Huang et al. addressed the question of whether to pre-condition CAD by replacing the equations present with their Gröbner basis [57]. They generated 1200 random 3-variable problems, with moderate bounds on degree and coefficient size. While they could train an ML model to address the question, there is no guarantee that real application problems will behave as the random data casting doubt over the generalisation ability. Synthetic datasets were also used in Refs. [28], [17], and [49], with the latter selecting the polynomial degrees, coefficients and number of terms from distributions whose mean and standard deviation matched the SMT-LIB dataset, to give greater chance of generalisability. However, of those three only [17] evaluated also on real-world data, which should be an essential step when using synthetic data.

We note that although random polynomial generation is cheap, synthetic data generation is still expensive if it needs to be labelled with CA for supervised learning. As a result, none of the datasets discussed so far have been large enough to support deep learning methods. This was addressed recently by Chen et al. in 2024 who used a high performance computing system to build a new labelled synthetic dataset for CAD variable ordering [24] of more than 20,000 problems (which can be augmented to more than 300,000). Furthermore, these are all meaningful since for every

example the runtime of the worst ordering is at least twice that of the best. Aside from its use to study this problem, the dataset was also used in Ref. [24] to uncover bugs in CA systems!

In addition to these works, two further datasets have recently been introduced. The first focuses on real quantifier elimination problems [25], providing a benchmark for evaluating learning-based approaches in this setting. The second is an enhanced four-variable dataset designed for studying variable ordering selection in CAD [26]. Together, these datasets further expand the available resources for training and evaluating ML methods in CA.

3.2.6 Feature Engineered Supervised Learning. A wide variety of ML tools were used in the above literature, however, due to the lack of a large-scale dataset until 2024, the work above used a feature-engineered supervised learning approach.

The original work of Huang et al. [58], [57] used **support vector machines** (**SVM**), and did not test with any other ML models. SVMs are known to be sensitive to outliers and can become computationally expensive for larger datasets.

The 2019 work of Florescu and England experimented also with **K-nearest neighbour** (**KNN**), **Decision Tree** (**DT**), and **Multi-layer Perceptron** (**MLP**) classifiers [44], while the 2023 work of del Rıo and England included also XGBoost and the 2020 work of Chen et al. [28] and the 2024 work of John and Davenport [66] included also artificial neural networks. There is no common consensus on which of these models is best suited to the problem; it can vary with other factors. In Ref. [38] the KNN and SVM models performed better under regression than classification, while RF, MLP, and XGBoost performed worse for example.

These models all presume a human feature engineering approach. The work of Huang et al. [58], [57] used a very limited number (11 for 3-variable problems) of hand crafted features, seeded from the statistics used in the Brown heuristic [14]. Such a small number of features likely leads to under-fitting.

To overcome this issue Florescu and England proposed an automated feature generation method from polynomials systems in 2019 [47], again inspired by the type of operations used in Brown's heuristic but generating all possible statistics they can make (78 unique features for 3-variable problems; 7 times more than in previous studies like [44], [58]). The added features greatly improved classifier performance and this approach was taken up all subsequent work that used a feature-engineered architecture.

3.2.7 Reinforcement Learning and Graph Neural Networks. One limitation in the reported literature so far is that supervised learning relies on datasets, expensive to label with CA and potentially prone to over-fitting. A study in 2024 by Jia et al. was the first to use **Reinforcement Learning** (**RL**) for the selection of variable ordering for CAD [63], where an agent was rewarded for its choices based on the number of CAD cells they led to. Their approach allowed for a single ML model that could make decisions for systems of different numbers of variables, addressing another of the key shortcomings in the prior work. A Chinese language study in 2024 also used RL for CAD [64].

Aside the moving to RL, [63] was also notable as the first article to make use of graph neural networks for the CAD variable ordering problem. Graphs are formed from polynomial sets with nodes being variables and edges indicating that two variables appear in the same polynomial. Such graphs have been previously shown to allow for the maintenance of polynomial sparsity during CAD [77], and so this is a natural approach to try, although this approach was not reported as successful when used outside of the RL paradigm in Ref. [66].

The dataset used for training in Ref. [63] was synthetic, randomly generated polynomial sets. While the RL agents could significantly outperform other approaches on such data, their performance on the SMT-LIB was less dominant, indicating work still to be done to address generalisation.

3.2.8 Deep Learning. Recent research has explored the use of Transformer models to improve variable ordering selection in CAD [65]. Since generating labelled data requires performing exhaustive CAD computations for every possible ordering, and there is limited labelled data available, this work proposed a pre-training and fine-tuning framework. The authors first pre-trained Transformer models on six specialized tasks, which force the Transformer to internalize deep polynomial structures and algebraic behaviours. These tasks included predicting exponent structures, algebraic feature vectors, and properties related to projection factors and resultants, all of which are relevant to CAD variable ordering, but cheap to derive data for. The pre-trained models were then fine-tuned on the expensively labelled CAD variable ordering datasets to specialize the models in predicting effective orderings.

The study showed that this approach outperformed traditional heuristic methods on public CAD benchmark datasets, including those from Ref. [24]. The authors further introduced an enhanced four-variable dataset, Ref. [26], containing a significantly higher proportion of difficult instances with running times exceeding 100 seconds for training and evaluation. They also evaluated the approach on a separate SMT-based CAD benchmark dataset to show models trained on randomly generated polynomial can generalize to more realistic symbolic computation problems.

Although the pre-training and fine-tuning strategy improved CAD variable ordering prediction and achieved substantial runtime improvements over classical heuristics, the article also observed that some pre-training tasks related to deeper symbolic operations, particularly those involving projection factors and resultants, were significantly more difficult for the Transformer models to learn accurately. This suggests that while the models can learn useful structural representations of polynomial systems, they do not yet fully capture the deeper symbolic reasoning required for more advanced algebraic computations.

3.2.9 Explainable AI. There have been numerous studies reporting the efficacy of ML tools for making the heuristic choice of variable ordering in CAD. However, the studies do not give any insight into why these ML tools are successfully making this choice. In 2024, Pickering et al. utilised the **explainable AI** (**XAI**) tool SHAP to gain insight into how the feature-engineered models were making their decisions [89]. SHAP analyses the impact of individual features, putting a number on their contribution to the overall decision [79] and is perhaps the most widely used XAI tool. In Ref. [89] the authors aggregated these scores across the dataset and had models vote to produce an overall ranking of the automated features develop in Ref. [47]. The top ranked features included some statistics already seen in the literature [14], [37] but also others that were new.

As a final experiment in Ref. [89], the authors sought to build a simple heuristic from three of the features, equivalent in complexity to the widely used Brown's heuristic [14]. They studied all 120 possible triples and found the best, named T1, to consist of the top three ranked features by SHAP. This suggest a piece of simple code (not involving any ML architecture) that could be easily swapped for Brown's heuristic in any CA system. The code is informed by XAI, but not data-dependent and of a complexity equivalent to a simple human-designed heuristic. In Ref. [24], a separate group tested T1 on an independent dataset and found strong performance, suggesting that XAI may be used to strip out complexity that leads to over-fitting, allowing for better generalisation between datasets.

Another XAI experiment in 2024 was conducted by Florescu and England [50]. Here the authors tried to train an inherently interpretable model (rather than explaining a complex one). They also sought something of the complexity of Brown's heuristic, and so modelled this heuristic's behaviour with a small neural network and then used ML training to improve the networks performance. The trained network could then be reinterpreted to code similar to Brown's heuristic. This approach trained the network on random data but evaluated it on the SMT-LIB dataset, again demonstrating that a simpler, XAI informed, model can allow for better generalisation between datasets.

3.2.10 Limitations. This area of symbolic computation has seen the most study with ML methods to date, although it has been focussed on the variable ordering choice with the other heuristic decisions receiving very limited attention. As noted in Ref. [59], additional decisions also bring in the question of which order to make the decisions in: if taken together then the possibilities grow substantially. There has been no study yet in the literature to propose a methodology for this.

There is also clearly a need for more work on the reinforcement learning approach, where there has been only two studies to date: since one of those studies also introduced the graph bases models we need an ablation study to understand how each innovation contributed to the performance gain. We note that when considered the choice between regression and classification in Ref. [38] the optimal choice of model and architecture changed and a deeper study is needed to understand what is optimal overall.

There has been substantial lessons learned over the use of datasets in this area, with the recent contribution in Ref. [24] opening the doorway to deep learning research. But there still remains unanswered questions over how well random data represents real-world applications, and whether generalisation away from the training dataset is possible. The recent work with XAI methods suggests one route to offer this, but only two studies have been done to date and there is a need to both replicate the hypothesis and explore the wider range of XAI applications.

4 Gröbner Basis Computation

Systems of non-linear polynomial equations appear in many areas of engineering, medicine and computer science. It is necessary to efficiently solve such systems for cryptography [96], robotics [1], physics [82] and other fields. One of the main tools widely used to solve non-linear polynomial systems is the computation of a Gröbner basis. A Gröbner basis is a generating set of an ideal over a closed ring of multivariate polynomials [55].

Although ideals are infinite sets, they are generated by a finite "generating set" of polynomials. Interestingly, the same ideal can have different generating sets, which can reproduce the original generating polynomials through linear combinations. Some generating sets are more convenient than others: a Gröbner Basis is a special generating set with advantageous properties. A Gröbner Basis may be used to simplifies many computations and extract properties of an ideal. It is particularly useful for solving systems of simultaneous equations represented by the ideal, making it a powerful tool in CA. Consider the example in Figure 3: the intersection of the four polynomials in the input is the same as the intersection of the three in the Gröbner Basis, but the latter uses much simpler polynomials.

The most widely used algorithms for computing a Gröbner basis are Buchberger's algorithm (introduced in his 1965 PhD thesis [18] and reproduced in English in Ref. [19]), and Faugére's F4 and F5 algorithms introduced in Refs. [45] and [46], respectively. Both algorithms have doubly exponential complexity in the worst case scenarios [83], but the F5 algorithm is generally considered to be the fastest. However, Buchberger's algorithm is simpler and easier to understand: the interpretability of that algorithm makes it a good choice for initial experiments on ML optimisation for improving Gröbner basis efficacy.

The ML work in the literature to date has focused on optimising the internals of Buchberger's algorithm and so we briefly introduce this. The algorithm begins with a generating set of polynomials for an ideal. From this, a set S-pairs is initialised by all pairs from the input set. Each time the main algorithm loop selects one of these, removing it from the set, with the loop terminating once the set is empty. The loop body computes the corresponding S-polynomial by cancelling leading terms and reducing this with respect to the basis. If the polynomial does not reduce to zero, then it is added to the basis and used to generate new S-pairs in the set by pairing it with the other basis members. Pseudocode and an example for Buchberger's algorithm can be found in Appendix A.

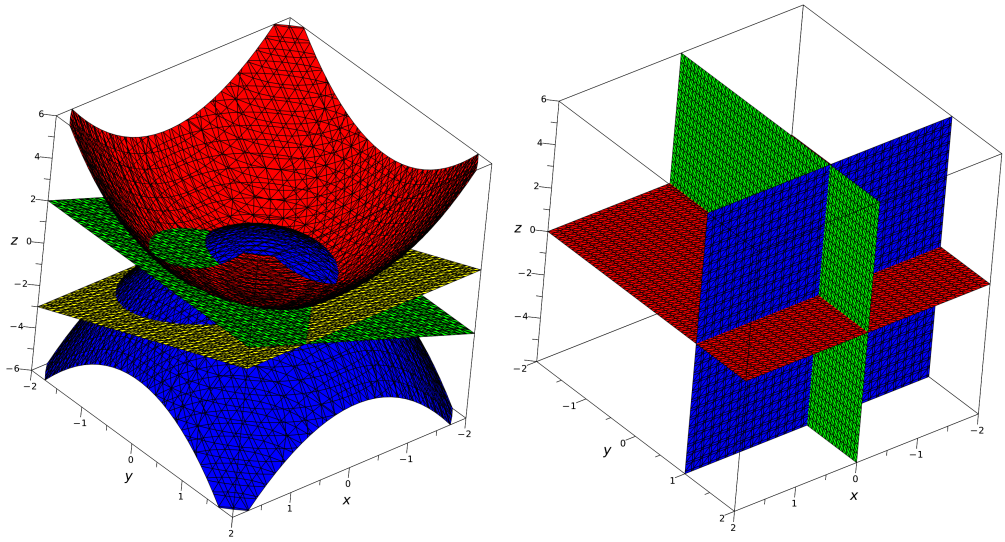

Fig. 3. An example in three dimensions (following [93]). Let $F = \{z - x^2 - y^2 + 1, z + x^2 + y^2 - 1, z - x, z - y + 1\}$ (as graphed on the left); then a Gröbner basis for F (under a lexicographical monomial ordering) is $\{z, y - 1, x\}$ (as graphed on the right). Note the curves on the left and right intersect at the same points.

4.1 Heuristic Decisions within Gröbner basis Computation

When computing a Gröbner basis there are choices that do not affect the correctness of the end result, but can impact how that result is presented and the resources used to obtain in. First, like CAD, there is a choice of variable ordering needed, however, a Gröbner basis is also computed relative to a monomial ordering. It is widely thought that some monomial orders are more efficient to compute with than others: leading researchers to study algorithms to convert bases from one ordering to another [29], [102]. However, it has been observed that these general principles do not always apply to specific application domains [95].

A third decision that can affect Buchberger's algorithm is the order in which calculations are performed within, specifically the order in which S-pairs are reduced (as defined shortly). Unlike the other two decisions this is not a single choice made one per algorithm call but a choice that needs to be made at the start of each iteration of the main algorithm loop. The alternative GB algorithms also have heuristic choices in their calculation order worthy similar study: the F4 algorithm uses linear algebra techniques to reduce large numbers of S-pairs at the same time, but the batching still requires analogous selection strategies; and although the F5 algorithm selects pairs in order of signature it introduces a new sensitivity to the order in which the polynomials are presented (see e.g., Ref. [31]).

Work on these decisions to date (both ML and otherwise) has focussed on them individually, but of course, they can be taken in tandem: a classic work of Czapor described how while Gröbner basis with lex monomial orderings are hard in general, the problems can be mitigated through a good choice of S-polynomial selection [32].

The choice of which S-pair to process next is critical for the computational complexity of the algorithm. Computation of S-pairs in a non-efficient sequence can lead to an increase in both the basis size, the number of loop iterations used, and the runtime. There are well established selection strategies (heuristics) to make the choice, such as the sugar selection strategy [51], but none of these is always dominant over the others (see e.g., Section 4.3 of Ref. [88]). The development of

specialised strategies for particular applications remains a topic of current research [75]. Given the prominence of this choice for the efficiency of the algorithm it has been viewed as a important candidate for ML optimisation.

4.2 Machine Learning for Making Heuristic Choices for Gröbner Bases

ML has been implemented for selection of certain heuristics as well as end to end computation of Gröbner basis. A summary of all the relevant literature identified is given in Table 4 in Appendix B. We discuss this thematically in the following subsections.

4.2.1 Learning Selection Strategies for Buchberger's Algorithm. In 2020, Peifer et al. [87] was the first one to apply ML tools to S-pair selection in Buchberger's algorithm. They utilised RL in which an agent (a neural network) makes choices and then receives a *reward* to indicate how good the choice was — the reward is used to edit the weights in the neural network to improve the next choice. In this case the reward is the number of polynomial additions required to reduce the chosen S-pair.

They represent the current state (the current set of S-pairs) as a matrix, in which each row represents one pair by their exponent vectors (the powers of variables). Note that the coefficients are not considered. They restrict their study to binomial ideals, meaning the width of the input matrix is fixes to a manageable size. They also work in a modular field to avoid coefficient growth, and consider only the grevlex monomial ordering. Thus there is much work still to be done in generalising their approach to the full scope of Gröbner basis calculations in applications.

They both train and test with only random data. They consider a variety of different distributions of random polynomials based on number of variables, number of generators, and degree. Their results suggest that agents trained on one distribution can perform also on other *close* distributions.

Later in 2021, Mojsilovic et al. used a similar setup to study the different parameters such as statistics of the input polynomial system and commutative algebra invariants derived from the ideal generators that can contribute towards the computational complexity of computing Gröbner basis using Buchberger's algorithm [84]. This time they seek to predict the number of polynomial additions a choice will require (i.e., the reward in Ref. [87]). The authors try first to use a multiple linear regression model built from a set of easy-to-compute statistics of the generator set, and later a recursive neural network both on these features and the polynomials themselves. They work only with binomial and toric ideals.

4.2.2 Learning Variable Ordering. Xu et al. in 2019 were concerned with the computational efficiency of Gröbner basis for geometric vision problems [111]. Here, solvers prepare fixed *elimination templates* to calculate the Gröbner basis online: they are concerned with selecting the variable and monomial ordering for a problem which leads to different templates. They restrict the study to the case of choosing between different permutations of templates and train a deep learning model for the task. Their model is trained on the basis of coefficients of polynomials to select the most efficient variable ordering, prioritising numerically stable results. They train on synthetic data but evaluate on vision problems.

4.2.3 Border Basis Optimisation. A border basis is a generalisation of Gröbner basis for the case of 0-dimensional ideals (defining a finite number of points) which is particularly useful in numerical polynomial algebra as it has better numerical stability compared with Gröbner bases [68]. Kera et al. [71] used transformers to select expansion directions in the algorithm to compute these; a similar choice to S-pair selection in Ref. [87], in that it needs to be made in each iteration of the main loop and the wrong choice can increase the runtime (but termination and output correctness is still guaranteed). The transformer's choices led to speed-up factors of 3.5× compared with the base algorithm [68].

4.3 Learning to Compute Gröbner Bases Directly

In 2024, Kera et al. utilised transformers to directly compute Gröbner bases, showing the potential for transformers to directly address exponential complexity problems [70]. Unlike [87] this study is not restricted to binomials. but it does only consider 0-dimensional radical ideals.

The article focuses on the problem of data generation: the power of transformers can only be utilised with large datasets whose production then becomes a CA problem in its own right. They consider the random generation of a Gröbner basis and then approaches to reverse engineer this into another generating set of the same ideal in order to produce pairs of input and output for a Gröbner basis algorithm. There remains work to do here since the data produced is quite biased in the relative lengths of the generator sets.

Another innovation in this article is a new hybrid embedding of the inputs to overcome the problem of dimensionality: rather than have numerous number tokens (increasing the embedding matrix size) or splitting numbers into digits (increasing the attention heads), the coefficients go through a separate embedding network which encodes a continuity bias (two close numbers are close in the embedding). The rest of the monomial (the variables and their powers) are stored as individual tokens, a different approach to the work in Ref. [71] for border bases where each monomial was a single token. Of course, all the data sampled still comes from fixed ranges.

Of course, the main limitation of this approach is that the results produced are not guaranteed to be correct, unlike those produced by symbolic computation methods like Buchberger's algorithm. Note that although it is easy to check whether a given generator set is a Gröbner Basis it is not easy to check whether it is the Gröbner Basis for the original set. This work requires new corresponding CA research to more efficiently check the validity of the transformer's answer. Note that the group behind this article have now released a Python library, CALT, designed to help non-experts in deep learning train models for symbolic computation tasks [69].

Building on [70] a new study [81] utilised hierarchical attention transformers to compute the the Gröbner Basis of multivariate polynomials. The tree-structured inductive bias better matches the hierarchical structure of polynomial systems and offers computational savings as compared with flat attention models. Furthermore by using architecture combined with curriculum learning (training the model on datasets with progressively increasing problem sizes), the authors were able to solve problem systems that were much larger (13 variables) compared with the ones reported in [70] (5 variables). The study also provided evidence for generalizability to arbitrary tree depths.

4.4 Learning Optimal Input for Gröbner Bases

In 2020 a study demonstrated the use of ML tools to approximate the relations of highly complex polynomial systems with the aim of creating simpler problems for study with a Gröbner basis [80]. The context is modelling energy systems with many actors and finding Nash equilibria and this approach makes the proposed methods scalable to much more complex systems. However, the approximations might not be able to fully capture the complexity of the original systems. Although the approach does not aim on optimising Gröbner basis computation itself, it may be possible for the same techniques to be used to optimise the form of the input generating set in other contexts.

4.5 Dataset Challenges

One aspect that becomes apparent in the studies above is the lack of a benchmark datasets for Gröbner basis construction. Without these, it is difficult to establish if any ML methodology is generalizable. The key studies above mostly both train and test on random data from the same generators, giving limited assurance against over-fitting. While the training and test sets are disjoint, when both are drawn from the same random generator they thus share the same underlying distribution and structural characteristics. Only a handful of papers partially offset this by experimenting

with different distributions in the data generators. We note the efforts of the Frisco, PoSSo and SymbolicData projects in the past [52], the PHCPack database [104], and the more recent ODEbase [78]. Nevertheless, a large-scale agreed community benchmark database for Gröbner basis, similar to the SMT-LIB would be a great asset.

4.6 Explainable AI

There has not yet been any application of XAI tools for Gröbner bases in the literature. Peifer et al. [87] did offer interpretations of their agents' strategy for the choice of S-pairs which seemed to have been obtained through manual analysis (see the thesis [88] for details). These revealed some preferences not previous considered in the literature: preferences for pairs that had monomials, and for those whose S-polynomial had lower degree. Used alone, these preferences perform better than the previous strategies in the literature, suggesting the benefit of constructing simple heuristics from an XAI analysis as found earlier for CAD.

Ref. [84] studied the contribution of complex algebraic features as well as simple statistics in their analysis to predict the number of polynomial additions performed. Surprisingly, they found the simple statistics to outperform the more expensive algebraic invariants in terms of predictive power: this analysis is claimed to disprove folklore on the importance of certain invariants. An XAI feature ranking approach might be able to add further insight on exactly which combinations of features are needed to make these predictions.

5 Symbolic Integration

A summary of all the relevant literature identified is given in Table 5 in Appendix B. We discuss this thematically in the following sections.

5.1 Deep Learning to Compute Symbolic Integrals

In 2020, Lample and Charton were the first to use transformers to directly undertake symbolic computations in Ref. [76], where they used transformers to perform symbolic indefinite integration and to solve ordinary differential equations. Their results demonstrate that not only can transformers undertake such tasks with high accuracy, they can also tackle problems for which traditional CA systems cannot (at least within the time limit given: the experimental results report together failure due to timeout and production of the wrong answer, but the latter should come only from the transformer). In many cases it is easy to check by differentiating but this is not always easy or possible and depends on the existing of high quality simplification tools.

In Ref. [76] the authors used sequence-to-sequence model, meaning both the input and output were viewed as a sequence of tokens (either number, variables, operators, or functions). They constructed these sequences in prefix (Polish) notation for efficiency (avoiding brackets). However, a subsequent work by Kubota et al. noted that prefix notation is not well suited to transformers since the operators and their subjects are not necessarily close to each other [72]. They experimented with other models and concluded that LSTMs, which learn the relative positions of tokens in the input, as superior for this task.

5.2 Dataset Issues

The work in Ref. [76] was replicated in a 2023 study [4] but with narrower constraints for data generation and a larger dataset.

The data is Ref. [76] is generated from random expressions which are then either differentiated or integrated (using an existing CA system) to form training pairs. Inadequacies in this dataset were the centrepiece of later critiques of Ref. [76]. Davis was the first to point out the pairs shared patterns in comparative length [35], with the dataset omitting pairs of roughly equal lengths.

This issue was addressed later in the works of Barket et al. [7], [8]. They proposed new data generation techniques for symbolic integration inspired by classical CA work for the Risch [92] and Risch-Norman algorithms [86], respectively.

Another concern with the dataset in Ref. [76] is that the training data comes from the same generation methods of the testing data. The authors did report how models trained on data from one generator performed when tested on another and identified generalisation issues. In the extended version of the article [90] the author's analysis of the dataset in Ref. [76] reveals that many of their test expressions (up to 74%) are very similar, meaning they differ only a constant or sign, to the training data. This particularly calls into question the appropriateness of the testing methodology.

Concerns may be partially addressed by the additional data generators in Refs. [7], [8] and by merging similar examples, but ideally we want an independent validation dataset. Once again, there are not agreed community benchmarks in use for symbolic integration. However, in this case, there does exist a large ($> 72k$) set of problems produced for the RUBI system [91]. The work in Ref. [9] also made use of a large-scale validation set used by Maplesoft as unit tests for the Maple CA system.

5.3 Learning for Guiding Symbolic Integration Algorithms

In 2024, Barket et al. researched the problem of choosing a suitable method for indefinite integration of a expression in the CA system Maple [9]. There were 12 methods available and the current Maple user-level command tries them in the same fixed order until one succeeds, in some cases with simple *guard* code to check applicability. This wastes runtime, but can also lead to unnecessarily complicated output: different algorithms can produce different (but mathematically equivalent) integrands. Barket et al. tackled this as a ML classification problem to select the sub-algorithm that produces the output with smallest size under Maple's data storage format for expressions. They experimented with **long short-term memory neural networks** (**LSTMs**) for the task. They found performance improved when using LSTMs for tree structures rather than just simple sequences, perhaps unsurprising given mathematical expressions naturally form trees, but a technique not used before in the literature. They trained and tested on the data generators in Refs. [76], [7], and [8]. They also evaluated on an independent dataset which showed issues of generalisation still to be addressed.

A later article by Barket et al. in 2024 looked at the guard code mentioned earlier: these are simple pieces of code that can identify the applicability of an integration method for an integrand that exist for 4 of the 12 methods in Maple [10]. They trained transformers to make binary predictions on a methods applicability: these proved successful in generating guards for methods without one, and more accurate that the existing guards for methods which had them.

In 2024, Sharma et al. considered a similar problem of ML guided integration, however, in their case they sought to apply rules to perform integration step-by-step as a student would by hand [97]. They created a dataset containing 2 million pairs of mathematical functions, and integration rules that can be applied to them. They use this to train a neural network to guide the choice of integration steps. There is no risk from incorrect results: if the network suggests an inapplicable integration rule then this is identified. They report that this leads to 6× fewer steps when integrated into SymPy's `integral_steps` function. The data was generated by runs of the original function, but the model can now guide the function to tackle problems it could not before, suggesting some clear generalisation has occurred.

5.4 Frontier LLMs Applied to Symbolic Integration

Increasingly we see frontier models, i.e., LLM based AI systems such as ChatGPT, Claude, DeepSeek, evaluated on mathematics problems. The MATH dataset included symbolic integration as a category

[54], and became a focus for LLM research, see e.g., Ref. [40], particularly for so-called reasoning models following a chain-of-thought process [109]. The MATH dataset contains step-by-step solutions for its problems, and so this is more like the task of Sharma et al. [97] than the direct symbolic computation of Lample and Charton [76]. However, unlike [97] the LLMs directly produce the steps without validation that they are correct.

5.5 New Symbolic-Integration Method Utilising Spare Regression

Irvanian et al., added symbolic integration functionality to the SciML package for Julia via a hybrid algorithm that included some learning techniques [62], [61]. They were inspired by the Risch-Norman algorithm [86] to generate an ansatz with multiple candidate terms for the integral (which they do with a combination of table lookup rules and algebraic manipulations). They then filter the candidate terms and apply sparse regression to find the coefficients. The approach has the benefit of finding integrals in a form similar to the provided integrand (what users likely expect).

5.6 Explainable AI

There has been very little use of Explainable AI in this area. Note that although [97] produces a method of integration that is interpretable to the user, unlike say the work of [76], the actual ML methods used are not explainable, i.e., we do not know why the neural networks makes the recommendation of integration rule that it does.

The only article to use an XAI method was [10] who used **layered integrated gradients (LIGs)** to analyse the transformer guards for integration methods. LIGs is a model specific post-hoc explainability technique providing local explanations [99] by highlighting the importance of individual input features (tokens in this case). For the most part this did not explain the transformers in Ref. [10] (where of course it is likely combinations of tokens that need to be analysed) but this did reveal one interesting finding: a particular function token which heavily contributed to predictions that a method should not be used. Upon inspection it was discovered that this function is indeed not in scope of this method: the XAI essentially revealed a shortcoming in the existing guard code which could be easily patched in the future.

6 Other Applications of ML in CA

A summary of the remaining relevant literature that we identified is given in Table 6. This includes some work similar to that we have seen in the previous sections but for different applications, which we summarise in the first few sections below. We also finish by noting some of work that does not directly perform symbolic computation but nevertheless may have useful lessons to learn from.

6.1 Using ML to Guide Symbolic Algorithms

In 2023 Berczi et al. [12] considered how ML can help with the resolution of singular points, where we replace such points in the solution set of a system of polynomial equations with smooth ones whilst keeping the rest of the solution set unchanged. Resolutions are not unique: the usual way to describe them involves repeatedly performing a fundamental operation known as *blowing-up*, with the complexity of the resolution highly dependent on certain choices. They translated the problem into a two-player game such that a winning strategy for the first play provides a solution to the resolution problem. They use reinforcement learning to train an agent to win the game and thus find the optimal resolutions. The agents trained outperform state-of-the-art heuristics. The reward is calculated as the total number of polynomial additions performed, similar to the work of Peifer et al. for optimising Buchberger's algorithm [87].

ML has recently been used to guide coordinate transformations for polynomial ideals: [53] considers the problem of putting ideals into quasi-stable position, which is a special coordinate system that retains many desirable properties of the generic initial ideal while being deterministically computable and verifiable. The algorithm which exists for this is under-specified: it needs to make a choice of *elementary move* to apply with any choice acceptable in terms of correctness and termination but giving different levels of efficiency. Traditional methods to make this decision rely either on random coordinate changes, which destroy sparsity properties, or simple human-designed metrics, which do not always yield optimal results. The authors explore a data-driven approach based on supervised ML with feature vectors of simple statistics about the ideal. Their experimental results demonstrate that trained models frequently identify nearly optimal transformations, preserving structure and improving computational efficiency. However, they both test and train with random ideals which, as they acknowledge, leaves open questions of generalisability.

6.2 ML Algorithm Selection for Resultants

We saw earlier studies on integration algorithm selection for particular problem instances [9], [10]. A similar study was carried out earlier for selecting among different algorithms to compute the resultant of two multivariate polynomials [98]. Simpson et al. used a feature engineering supervised learning approach for choosing from the four possible choices in Maple and Mathematica. They experimented with a variety of ML methods, finding neural networks performed the best: they reported 68% runtime improvement in Maple and 49% improvement in Mathematica. This study is a proof how ML can be used to make traditional CA systems more efficient. Furthermore, investigating these black-box models using XAI might result in new insights for making this choice and corresponding new algorithm development.

6.3 ML to Improve CA for Physics

There is a long history of dedicated CA systems for physics calculations, often tailored to performing a small number of calculations many times at high speed. These systems are highly optimised and so early adopters in the use of ML methods for CA optimisation. A key optimisation that has been studied in Refs. [73], [94], [74] is to construct polynomials in a form optimised for speedy numerical evaluation, by optimising the order of variables used in Horner factorisation. The authors use variants of Monte Carlo Tree Search to learn this, including in combination with simulated annealing, for the specialised CA system FORM.

More recently, Alnuqaydan et al. studied how ML can replace the symbolic computation in the computation of squared amplitudes, a key task in high energy physics calculations [5]. The authors use a sequence-to-sequence transformer model to replace this most expensive part of the symbolic computation: they report that ML is able to predict this correctly 97−99% of the time, at a speed that is up to two orders of magnitude faster than the symbolic computation. It is not clear what the dangers are to the wider physics experiments from the small number of error cases of the ML.

6.4 Linear Polynomial Equation Solving

We are mostly concerned with non-linear polynomial systems, but note that linear programming is also being addressed with ML tools. Of particular interest is the article of Dabelow and Ueda in 2024 which proposed a methodology to solve linear polynomial equations using deep neural networks and a reinforcement learning paradigm [33]. They utilise a symbolic stack calculator that applies different transformations to solve the equations. The RL agent can suggest actions, such as applying mathematical operations. The framework ensures that poor choices from the agent do not lead to incorrect mathematics (although they may not lead to the solution of the problem). This framework does not make use of existing CA tools: if it had then it is possible these would have

allowed it to improve performance, but on the other hand that would have limited it to strategies already designed by humans. They emphasise that the solutions provided are comprehensible to humans and step-by-step (although like [97] the decisions of the actual agent are not themselves explained). This work sits in contrast to approaches like [105] which seek to train transformers to solve systems of linear equations directly.

6.5 Learning from the MATH-AI Community

As mentioned earlier for integration, we increasingly see frontier LLM models applied to mathematics [54], [40], including some tasks traditionally thought of as symbolic computation. A survey in 2022 on performance of LLMs on basic arithmetic concludes that state-of-the-art LLMs fell short when probed with relatively simple tasks [101]. However, another survey in 2025 [106] demonstrated substantial improvements in the mathematical reasoning capabilities of LLMs through approaches such as Chain-of-Thought reasoning, reinforcement learning, test-time scaling, and knowledge-augmented reasoning. Despite these developments, current state-of-the-art models still face important challenges related to reasoning efficiency, exploration diversity, optimality of reasoning paths, and robust generalization to more diverse domains [106].

Success in more advanced mathematics has come from more specialised architectures, such as AlphaGeometry which tackled Olympiad problems through a combination of reinforcement learning a deep neural network and a symbolic deduction engine [103]; or its sister project AlphaProof which generates formally verifiable mathematical proofs in Lean, again using reinforcement learning with feedback from a check using Lean [60]. It has already been proposed that a similar joint approach is needed for symbolic computation and LLMs [110]. CA systems like Mathematica now integrate LLMs throughout their product, but there is not yet published research on the right way to optimise this.

There is a growing *Math-AI* community looking at specialised AI models for mathematical tasks. The focus here does not tend to be direct symbolic computation but the methodological developments may well have use for that. For example, the work using tree-embeddings for symbolic integration method selection described earlier [9] followed work using the tree-embeddings for mathematical formula search in Ref. [107] and for recognising the equivalence, derivative, or variable substitution between pairs of polynomials in Ref. [108], which may be useful for symbolic simplification. The latter stressed the importance of the synthetic data generation for their work and credited this with much of the improved performance. A related architectural development that used tree structures was reported in Ref. [81], where Gröbner bases were computed using Hierarchical Attention Transformers, where tree structure is built into the attention mechanism itself rather than just the input embedding, yielding substantial efficiency gains and scaling to larger polynomial systems.

6.6 Explainable AI

None of the above work made use of explainability for the actual ML. There is very limited work on XAI for LLMs, although we note the recent report from Anthropic: in particular, this seems to reveal that their LLM implements a rather non-standard addition algorithm for integer arithmetic [6]. Note also that the same report makes clear that when requested to explain its working, the LLM does not explain the actual algorithm seemingly used!

Charton has worked on explaining transformers designed for mathematics tasks [22, 23]. In a 2022 article, they investigated the failure cases of transformers trained on matrix inversion and eigenvalue decomposition [22]. He showed that incorrect model predictions still retain some mathematical properties of the solution: while models gave the right matrix 92% of the time, in 99% of examples the the matrix given had eigenvalues with less than 1% error in the L_1 norm. The

conjecture arising from this is that mathematics transformers do not hallucinate absurd solutions. It also gives lessons on the dataset used: a smaller set with wider range of eigenvalues gives better generalisation. Later, in his 2024 article concerning a transformer model trained to compute the greatest common divisor, he also looked at explaining the performance [23]. The approach to explainability here was to engineer experiments to test hypothesis, rather than using any tools based on model parameters or input features. This revealed some meaningful insights—such as that transformers learn to cluster input pairs with the same GCD and that early in training transformers learn to predict products of divisors of the base used to represent integers—and so this method of explainability may offer a promising direction for other applications.

7 Limitations and Future Possibilities of ML for CA

It is evident that ML tools have played a role in optimising CA algorithms. The reported studies suggest that the practical computational efficacy of computationally intensive algorithms such as CAD and the computation of Gröbner basis can be improved with the aid of ML tools, although care needs to be taken in the methodology used to achieve this. The variable ordering of CAD has become a well analysed case study in ML optimisation of CA, as described in Section 3. The Gröbner basis is perhaps the most commonly used CA tool for non-linear polynomials, but has only received study with ML more recently as described in Section 4, perhaps due to its heuristic choices being more embedded within the algorithm and thus more difficult to study. There is still scope for further work on both applications: for GB the state-of-the-art is limited in scope to certain ideal types, while for CAD there exists many heuristic decisions other that the variable ordering which have received little attention.

In Section 5, we saw that for symbolic integration metrics other than runtime were brought to the fore, with studies looking to optimise the form and interpretability of the output. Although a variety of other areas of CA were found to have been studied by ML in the literature in Section 6, it is clear that the applications that have been studied to date are only a small fraction of application areas within the field. The development of a shared methodology and toolkit would greatly aid developers to transfer knowledge between applications.

A common issue that arose is the inadequacy of benchmark dataset that can be used to make the proposed methodologies scalable and generalizable. It has long been accepted in CA that real-world applications data can look very different from the randomly generated datasets. Despite this, there exist few large-scale datasets and no community organisation controlling them. For CAD, researchers took advantage of the SMT-LIB developed and controlled by the satisfiability checking community. However, even this is not really of a scale for deep learning and it was found to need careful preparation before it can be used meaningfully. In the absence of large real-world datasets, it is inevitable that synthetic data will be used for training, but in that case it should become a community standard of rigour that models are evaluated on real data.

There does not exist clear conclusions on which ML tools are the most effective for symbolic computation. As with most ML applications, there needs to be a tuning stage to the application at hand. Recent work suggest the use of tree-based models [9] and reinforcement learning [63] as directions deserving future research.

There exists a sharp choice between the use of ML to perform symbolic computation directly and its use to to optimise CA systems in a way that does not risk output correctness, as exemplified by [87] and [70] for Gröbner basis computation. The latter will likely generate new CA problems in algorithms to validate ML suggest results, and both approaches generate new CA problems in high quality synthetic dataset generation. It also worth noting that mathematics problems like these are particularly interesting for ML: the difficulties in finding the right embedding and in identifying biases within datasets, make this a good testing ground for new ML innovations.

Another key finding that emerged during this survey is that there are relatively few studies that use ML to entirely replace CA algorithms. Existing examples include work on symbolic integration [72, 76] and Gröbner basis computation [70, 81] discussed earlier, but nothing else was identified.

We finish by noting that another area deserving future focus is the use of explainable AI. Most of the work surveyed gave no consideration to explainability but those that did suggested there is real value in doing so: both from new understanding of the algebra [87], from new understanding of how transformers learn [23], and from better generalisability arising from simpler analysis suggested by XAI [89]. By developing this and the other innovation described earlier, we may greatly increase the scope of applications for CA in real-world applications.

References

[1] Rafał Abłamowicz. 2010. *Some Applications of Gröbner Bases in Robotics and Engineering*. Springer London, London, 495–517. DOI: https://doi.org/10.1007/978-1-84996-108-0_23

[2] Erika Ábrahám, James H. Davenport, Matthew England, and Gereon Kremer. 2021. Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings. *Journal of Logical and Algebraic Methods in Programming* 119 (2021), 100633. DOI: https://doi.org/10.1016/j.jlamp.2020.100633

[3] Behzad Akbarpour and Lawrence C. Paulson. 2010. MetiTarski: An Automatic theorem prover for real-valued special functions. *Journal of Automated Reasoning* 44, 3 (2010), 175–205. DOI: https://doi.org/10.1007/s10817-009-9149-2

[4] Dhurgham Abbas Albojwaid, Saad Talib Hasson, and Mohammed Shaker Mahmood. 2023. Symbolic calculations for different datasets in python. In *Fundamental and Applied Scientific Research in the Development of Agriculture in the Far East (AFE-2022)*, Khasanov Sayidjakhon Zokirjon ugli, Aleksei Muratov, and Svetlana Ignateva (Eds.). Springer Nature Switzerland, 849–857. DOI: https://doi.org/10.1007/978-3-031-36960-5_96

[5] Abdulhakim Alnuqaydan, Sergei Gleyzer, and Harrison Prosper. 2023. SYMBA: Symbolic computation of squared amplitudes in high energy physics with machine learning. *Machine Learning: Science and Technology* 4, 1 (Jan 2023), 015007. DOI: https://doi.org/10.1088/2632-2153/acb2b2

[6] Anthropic. 2025. *Tracing the thoughts of a large language model*. Technical Report. Anthropic. Retrieved from https://www.anthropic.com/research/tracing-thoughts-language-model

[7] Rashid Barket, Matthew England, and Jürgen Gerhard. 2023. Generating elementary integrable expressions. In *Computer Algebra in Scientific Computing*, François Boulier, Matthew England, Ilias Kotsireas, Timur M. Sadykov, and Evgenii V. Vorozhtsov (Eds.). Springer Nature Switzerland, 21–38. DOI: https://doi.org/10.1007/978-3-031-41724-5_2

[8] Rashid Barket, Matthew England, and Jürgen Gerhard. 2024. The liouville generator forÂ producing integrable expressions. In *Computer Algebra in Scientific Computing*, François Boulier, Chenqi Mou, Timur M. Sadykov, and Evgenii V. Vorozhtsov (Eds.). Springer Nature Switzerland, Cham, 47–62. DOI: https://doi.org/10.1007/978-3-031-69070-9_4

[9] Rashid Barket, Matthew England, and Jürgen Gerhard. 2024. Symbolic Integration Algorithm Selection with Machine Learning: LSTMs Vs Tree LSTMs. In *Mathematical Software – ICMS 2024*, Kevin Buzzard, Alicia Dickenstein, Bettina Eick, Anton Leykin, and Yue Ren (Eds.). Springer Nature Switzerland, 167–175. DOI: https://doi.org/10.1007/978-3-031-64529-7_18

[10] Rashid Barket, Uzma Shafiq, Matthew England, and Juergen Gerhard. 2024. Transformers to predict the applicability of symbolic integration routines. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*. 10 pages. Retrieved from https://openreview.net/forum?id=b2Ni828As7

[11] Clark Barrett and Cesare Tinelli. 2018. *Satisfiability modulo theories*. Springer, Cham, 305–343. DOI: https://doi.org/10.1007/978-3-319-10575-8

[12] Gergely Berczi, Honglu Fan, and Mingcong Zeng. 2023. An ML approach to resolution of singularities. In *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) (Proceedings of Machine Learning Research, Vol. 221)*, Timothy Doster, Tegan Emerson, Henry Kvinge, Nina Miolane, Mathilde Papillon, Bastian Rieck, and Sophia Sanborn (Eds.). PMLR, 469–487. Retrieved from https://proceedings.mlr.press/v221/berczi23a.html

[13] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[14] Christopher W. Brown. 2004. Cylindrical Algebraic Decomposition. Companion to the Tutorial: Cylindrical Algebraic Decomposition, presented at ISSAC'04. Retrieved January 21, 2025 from https://www.usna.edu/Users/cs/wcbrown/research/ISSAC04/handout.pdf

[15] Christopher W. Brown. 2015. Open non-uniform cylindrical algebraic decompositions. In *Proceedings of the 2015 International Symposium on Symbolic and Algebraic Computation (ISSAC'15)*. ACM, 85–92. DOI: https://doi.org/10.1145/2755996.2756654

[16] Christopher W. Brown and James H. Davenport. 2007. The complexity of quantifier elimination and cylindrical algebraic decomposition. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation* (Waterloo, Ontario, Canada) *(ISSAC'07)*. Association for Computing Machinery, New York, NY, USA, 54–60. DOI: https://doi.org/10.1145/1277548.1277557

[17] Christopher W. Brown and Glenn C. Daves. 2020. Applying Machine Learning to Heuristics for Real Polynomial Constraint Solving. In *Mathematical Software – ICMS 2020 (Lecture Notes in Computer Science, Vol. 12097)*, A. Bigatti, J. Carette, J. H. Davenport, M. Joswig, and T. de Wolff (Eds.). Springer International Publishing, 292–301. DOI: https://doi.org/10.1007/978-3-030-52200-1_29

[18] Bruno Buchberger. 1965. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal. (An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal).* PhD Thesis. Mathematical Institute, University of Innsbruck.

[19] Bruno Buchberger. 2006. Bruno Buchberger's PhD Thesis (1965): An Algorithm for Finding the Basis elements of the residue class ring of a zero dimensional polynomial ideal. *Journal of Symbolic Computation* 41, 3–4 (2006), 475–511. DOI: https://doi.org/10.1016/j.jsc.2005.09.007

[20] Bruno Buchberger and Hoon Hong. 1991. *Speeding up Quantifier Elimination by Gröbner Bases.* Technical Report. 91-06. RISC, Johannes Kepler University. Retrieved from http://www3.risc.jku.at/publications/download/risc_1875/1991-02-08-A.pdf

[21] Bob Caviness and Jeremy Johnson. 1998. *Quantifier Elimination and Cylindrical Algebraic Decomposition.* Springer-Verlag. DOI: https://doi.org/10.1007/978-3-7091-9459-1

[22] François Charton. 2022. What is my Math Transformer doing? – Three Results on Interpretability and Generalization. In *Proc. 2nd MATH-AI Workshop at NeurIPS 2022.* 8 pages. DOI: https://doi.org/10.48550/arXiv.2211.00170

[23] Francois Charton. 2024. Learning the Greatest Common Divisor: Explaining Transformer Predictions. In *The Twelfth International Conference on Learning Representations (ICLR 2024).* 21 pages. Retrieved from https://openreview.net/forum?id=cmcD05NPKa

[24] Changbo Chen, Rui-Juan Jing, Chengrong Qian, Yaru Yuan, and Yuegang Zhao. 2024. A Dataset for Suggesting Variable Orderings for cylindrical algebraic decompositions. In *Computer Algebra in Scientific Computing*, François Boulier, Chenqi Mou, Timur M. Sadykov, and Evgenii V. Vorozhtsov (Eds.). Springer Nature Switzerland, 100–119. DOI: https://doi.org/10.1007/978-3-031-69070-9_7

[25] Changbo Chen, Rui-Juan Jing, and Yaru Yuan. 2026. A Dataset for Real Quantifier Elimination. DOI: https://doi.org/10.57760/sciencedb.31854

[26] Changbo Chen, Rui-Juan Jing, and Yuegang Zhao. 2025. An Enhanced Four Variables Dataset for Suggesting Variable Orderings for Cylindrical Algebraic Decompositions. DOI: https://doi.org/10.57760/sciencedb.31989

[27] Changbo Chen and Marc Moreno Maza. 2014. An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions. In *Computer Mathematics*, R. Feng, W. Lee, and Y. Sato (Eds.). Springer Berlin, 199—221. DOI: https://doi.org/10.1007/978-3-662-43799-5_17

[28] Changbo Chen, Zhangpeng Zhu, and Haoyu. Chi. 2020. Variable Ordering Selection for Cylindrical Algebraic Decomposition with Artificial Neural Networks. In *Mathematical Software – ICMS 2020 (Lecture Notes in Computer Science, Vol. 12097)*, Anna Bigatti, Jacques Carette, James H. Davenport, Michael Joswig, and Timo de Wolff (Eds.). Springer International Publishing, 281–291. DOI: https://doi.org/10.1007/978-3-030-52200-1_28

[29] Stéphane Collart, Michael Kalkbrener, and Daniel Mall. 1997. Converting Bases with the Gröbner Walk. *Journal of Symbolic Computation* 24, 3 (1997), 465–469. DOI: https://doi.org/10.1006/jsco.1996.0145

[30] George. E. Collins. 1975. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages.* Springer-Verlag (reprinted in the collection [21]), 134–183. DOI: https://doi.org/10.1007/3-540-07407-4_17

[31] David A. Cox, John Little, and Donal O'Shea. 2015. *Additional Gröbner Basis Algorithms.* Springer International Publishing, 539–591. DOI: https://doi.org/10.1007/978-3-319-16721-3_10

[32] Stephen R. Czapor. 1991. A Heuristic Selection Strategy for Lexicographic Groebner Bases?. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation (ISSAC'91).* ACM, 39–48. DOI: https://doi.org/10.1145/120694.120700

[33] Lennart Dabelow and Masahito Ueda. 2025. Symbolic Equation Solving via Reinforcement Learning. *Neurocomputing* 613 (2025), 128732. DOI: https://doi.org/10.1016/j.neucom.2024.128732

[34] James H. Davenport and Joos Heintz. 1988. Real Quantifier Elimination is Doubly Exponential. *Journal of Symbolic Computation* 5, 1-2 (1988), 29–35. DOI: https://doi.org/10.1016/S0747-7171(88)80004-X

[35] Ernest Davis. 2019. The Use of Deep Learning for Symbolic Integration: A review of (Lample and Charton, 2019). *Unpublished, available as arXiv preprint arXiv:1912.05752* (2019), 7 pages. https://arxiv.org/abs/1912.05752

[36] Tereso del Rio and Matthew England. 2023. Data Augmentation for Mathematical Objects. In *Proceedings of the 8th Workshop on Satisfiability Checking and Symbolic Computation (SC2'23).* CEUR Workshop Proceedings, Germany, 29–38. Retrieved from https://ceur-ws.org/Vol-3455/

[37] Tereso del Río and Matthew England. 2022. New Heuristic to Choose a Cylindrical Algebraic Decomposition Variable Ordering Motivated by Complexity Analysis. In *Computer Algebra in Scientific Computing*, François Boulier, Matthew England, Timur M. Sadykov, and Evgenii V. Vorozhtsov (Eds.). Springer International Publishing, United Kingdom, 300–317. DOI: https://doi.org/10.1007/978-3-031-14788-3_17

[38] Tereso del Río and Matthew England. 2024. Lessons on Datasets and Paradigms in Machine Learning for Symbolic Computation: A case study on CAD. *Mathematics in Computer Science* 18, 3 (2024), 17. DOI: https://doi.org/10.1007/s11786-024-00591-0

[39] Andreas Dolzmann, Andreas Seidl, and Thomas Sturm. 2004. Efficient Projection Orders for CAD. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation* (Santander, Spain) *(ISSAC'04)*. Association for Computing Machinery, New York, NY, USA, 111–118. DOI: https://doi.org/10.1145/1005285.1005303

[40] Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng et al. 2022. A Neural Network Solves, Explains, and Generates University Math Problems by Program Synthesis and Few-shot Learning at Human Level. *Proceedings of the National Academy of Sciences* 119, 32 (2022), e2123433119. DOI: https://doi.org/10.1073/pnas.2123433119

[41] Matthew. England. 2018. Machine Learning for Mathematical Software. In *Mathematical Software—Proc. ICMS 2018 (Lecture Notes in Computer Science, Vol. 10931)*, James H. Davenport, Manuel Kauers, George Labahn, and Josef Urban (Eds.). Springer International Publishing, 165–174. DOI: https://doi.org/10.1007/978-3-319-96418-8_20

[42] Matthew England. 2022. SC-Square: Future Progress with Machine Learning. In *Proceedings of the 6th Workshop on Satisfiability Checking and Symbolic Computation (*SC^2 *2022) (CEUR Workshop Proceedings, 3273)*, Curtis Bright and James H. Davenport (Eds.). CEUR-WS.org, 7–16. Retrieved from http://ceur-ws.org/Vol-3273/

[43] Matthew. England, Russell Bradford, Changbo Chen, James H. Davenport, Marc Moreno Maza, and David Wilson. 2014. Problem Formulation for Truth-table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition. In *Intelligent Computer Mathematics*, Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban (Eds.). Lecture Notes in Artificial Intelligence, Vol. 8543. Springer International, 45–60. DOI: https://doi.org/10.1007/978-3-319-08434-3_5

[44] Matthew England and Dorian Florescu. 2019. Comparing Machine Learning Models to Choose the Variable Ordering for Cylindrical Algebraic Decomposition. In *Intelligent Computer Mathematics*, Cezary Kaliszyk, Edwin Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen (Eds.). Springer International Publishing, 93–108. DOI: https://doi.org/10.1007/978-3-030-23250-4_7

[45] Jean C. Faugère. 1999. A New Efficient Algorithm for Computing Gröbner Bases (F4). *Journal of Pure and Applied Algebra* 139, 1-3 (1999), 61–88. DOI: https://doi.org/10.1016/S0022-4049(99)00005-5

[46] Jean C. Faugère. 2002. 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)*. ACM, 75–83. DOI: https://doi.org/10.1145/780506.780516

[47] Dorian Florescu and Matthew England. 2019. Algorithmically Generating New Algebraic Features of Polynomial Systems for Machine Learning. In *Proceedings of the 4th Workshop on Satisfiability Checking and Symbolic Computation (SC2'19) (CEUR Workshop Proceedings, Vol. 2460)*. 12 pages. Retrieved from http://ceur-ws.org/Vol-2460/

[48] Dorian Florescu and Matthew England. 2020. Improved Cross-Validation for Classifiers that Make Algorithmic Choices to Minimise Runtime Without Compromising Output Correctness. In *Mathematical Aspects of Computer and Information Sciences*, Daniel Slamanig, Elias Tsigaridas, and Zafeirakis Zafeirakopoulos (Eds.). Springer International Publishing, 341–356. DOI: https://doi.org/10.1007/978-3-030-43120-4_27

[49] Dorian Florescu and Matthew England. 2020. A Machine Learning based Software Pipeline to Pick the Variable Ordering for Algorithms with Polynomial Inputs. In *Mathematical Software – ICMS 2020 (Lecture Notes in Computer Science, Vol. 12097)*, Anna Bigatti, Jacques Carette, James H. Davenport, Michael Joswig, and Timo de Wolff (Eds.). Springer International Publishing, 302–322. DOI: https://doi.org/10.1007/978-3-030-52200-1_30

[50] Dorian Florescu and Matthew England. 2024. Constrained Neural Networks for Interpretable Heuristic Creation to Optimise Computer Algebra Systems. In *Mathematical Software ICMS 2024*, Kevin Buzzard, Alicia Dickenstein, Bettina Eick, Anton Leykin, and Yue Ren (Eds.). Springer Nature Switzerland, 186–195. DOI: https://doi.org/10.1007/978-3-031-64529-7_19

[51] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. 1991. One Sugar Cube, Please; or Selection Strategies in the Buchberger Algorithm. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation (ISSAC'91)*. ACM, 49–54. DOI: https://doi.org/10.1145/120694.120701

[52] Hans G. Gräbe. 2019. 20 Years SymbolicData. *ACM Commun. Comput. Algebra* 52, 3 (2019), 45–54. DOI: https://doi.org/10.1145/3313880.3313881

[53] Amir Hashemi, Mahshid Mirhashemi, and Werner M. Seiler. 2025. Machine Learning Parameter Systems, Noether Normalisations and Quasi-stable Positions. *Journal of Symbolic Computation* 126 (2025), 102345. DOI: https://doi.org/10.1016/j.jsc.2024.102345

[54] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving with the MATH Dataset. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 11 pages. Retrieved from https://openreview.net/forum?id=7Bywt2mQsCe

[55] Israel N. Herstein. 1975. *Topics in Algebra*. Wiley.

[56] John Hester, Briland Hitaj, Grant Passmore, Sam Owre, Natarajan Shankar, and Eric Yeh. 2023. An Augmented MetiTarski Dataset for Real Quantifier Elimination Using Machine Learning. In *Intelligent Computer Mathematics: 16th International Conference, CICM 2023, Cambridge, UK, , September 5–8, 2023 Proceedings* (Cambridge, United Kingdom). Springer-Verlag, Berlin, 297–302. DOI: https://doi.org/10.1007/978-3-031-42753-4_21

[57] Zongyan Huang, Matthew England, James H. Davenport, and Lawrence C. Paulson. 2016. Using Machine Learning to Decide when to Precondition Cylindrical Algebraic Decomposition with Groebner Bases. In *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. 45–52. DOI: https://doi.org/10.1109/SYNASC.2016.020

[58] Zongyan Huang, Matthew England, David Wilson, James H. Davenport, Lawrence C. Paulson, and James Bridge. 2014. Applying Machine Learning to the Problem of Choosing a Heuristic to Select the Variable Ordering for Cylindrical Algebraic Decomposition. In *Intelligent Computer Mathematics*, Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban (Eds.). Springer International Publishing, 92–107. DOI: https://doi.org/10.1007/978-3-319-08434-3_8

[59] Zongyan Huang, Matthew England, David J. Wilson, James Bridge, James H. Davenport, and Lawrence C. Paulson. 2019. Using Machine Learning to Improve Cylindrical Algebraic Decomposition. *Mathematics in Computer Science* 13, 4 (2019), 461–488. DOI: https://doi.org/10.1007/s11786-019-00394-8

[60] Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z. Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom et al. 2025. Olympiad-level formal mathematical reasoning with reinforcement Learning. *Nature* 651 (2025), 607–613. DOI: https://doi.org/10.1038/s41586-025-09833-y

[61] Shahriar Iravanian, Shashi Gowda, and Christopher Rackauckas. 2024. Hybrid Symbolic-Numeric and Numerically-Assisted Symbolic Integration. In *Proceedings of the 2024 International Symposium on Symbolic and Algebraic Computation* (Raleigh, NC, USA) *(ISSAC'24)*. Association for Computing Machinery, New York, NY, USA, 410–418. DOI: https://doi.org/10.1145/3666000.3669714

[62] Shahriar Iravanian, Carl Julius Martensen, Alessandro Cheli, Shashi Gowda, Anand Jain, Yingbo Ma, and Chris Rackauckas. 2022. Symbolic-numeric integration of univariate expressions based on sparse regression. *ACM Commun. Comput. Algebra* 56, 2 (2022), 84–87. DOI: https://doi.org/10.1145/3572867.3572882

[63] Fuqi Jia, Yuhang Dong, Minghao Liu, Pei Huang, Feifei Ma, and Jian Zhang. 2024. Suggesting Variable Order for Cylindrical Algebraic Decomposition via Reinforcement Learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (Red Hook, NY, USA) *(NIPS'23)*. Curran Associates Inc., 76098–76119. Retrieved from https://openreview.net/forum?id=vNsdFwjPtL

[64] Ruijuan Jing, Chengrong Qian, and Changbo Chen. 2024. Variable Ordering Selection for Cylindrical Algebraic Decomposition via Reinforcement Learning. *Journal of Systems Science and Mathematical Sciences* 44, 9 (2024), 2826–2849. DOI: https://doi.org/10.12341/jssms22799

[65] Rui-Juan Jing, Yuegang Zhao, and Changbo Chen. 2026. Breaking the Data Barrier in Learning Symbolic Computation: A Case Study on Variable Ordering Suggestion for Cylindrical Algebraic Decomposition. arXiv:2601.13731. DOI: https://doi.org/10.48550/arXiv.2601.13731

[66] Rohit John and James Davenport. 2024. Exploring Alternative Machine Learning Models for Variable Ordering in Cylindrical Algebraic Decomposition. In *Mathematical Software – ICMS 2024: 8th International Conference, Durham, UK, July 22–25, 2024, Proceedings* (Berlin). Springer-Verlag, 176–185. DOI: https://doi.org/10.1007/978-3-031-64529-7_20

[67] Dejan Jovanovic and Leonardo de Moura. 2012. Solving Non-Linear Arithmetic. In *Automated Reasoning: 6th International Joint Conference (IJCAR)*, Bernhard Gramlich, Dale Miller, and Uli Sattler (Eds.). Lecture Notes in Computer Science, Vol. 7364. Springer, 339–354. DOI: https://doi.org/10.1007/978-3-642-31365-3_27

[68] Achim Kehrein and Martin Kreuzer. 2006. Computing border bases. *Journal of Pure and Applied Algebra* 205, 2 (2006), 279–295. DOI: https://doi.org/10.1016/j.jpaa.2005.07.006

[69] Hiroshi Kera, Shun Arakawa, and Yuta Sato. 2025. CALT: A Library for Computer Algebra with Transformer. *arXiv preprint arXiv:2506.08600* (2025), 8 pages. DOI: https://doi.org/10.48550/arXiv.2506.08600

[70] Hiroshi Kera, Yuki Ishihara, Yuta Kambe, Tristan Vaccon, and Kazuhiro Yokoyama. 2024. Learning to Compute Gröbner Bases. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*. 27 pages. Retrieved from https://openreview.net/forum?id=ZRz7XlxBzQ

[71] Hiroshi Kera, Nico Pelleriti, Yuki Ishihara, Max Zimmer, and Sebastian Pokutta. 2025. Computational Algebra with Attention: Transformer Oracles for Border Basis Algorithms. DOI: https://doi.org/10.48550/arXiv.2505.23696

[72] Hazumi Kubota, Yuta Tokuoka, Takahiro G. Yamada, and Akira Funahashi. 2022. Symbolic Integration by Integrating Learning Models With Different Strengths and Weaknesses. *IEEE Access* 10 (2022), 47000–47010. DOI: https://doi.org/10.1109/ACCESS.2022.3171329

[73] Jan Kuipers, Aske Plaat, Jozef Antoon M. Vermaseren, and H. Jaap van den Herik. 2013. Improving multivariate Horner schemes with Monte Carlo tree search. *Computer Physics Communications* 184, 11 (2013), 2391–2395. DOI: https://doi.org/10.1016/j.cpc.2013.05.008

[74] Jan Kuipers, Takahiro Ueda, and Jozef Antoon M. Vermaseren. 2015. Code optimization in FORM. *Computer Physics Communications* 189 (2015), 1–19. DOI: https://doi.org/10.1016/j.cpc.2014.08.008

[75] Takashi Kurokawa, Takuma Ito, Naoyuki Shinohara, Akihiro Yamamura, and Shigenori Uchiyama. 2023. Selection Strategy of F4-style Algorithm to Solve MQ Problems Related to MPKC. *Cryptography* 7, 1 (2023), article number 10. DOI: https://doi.org/10.3390/cryptography7010010

[76] Guillaume Lample and François Charton. 2020. Deep Learning for Symbolic Mathematics. In *International Conference on Learning Representations (ICLR 2020)*. 24 pages. Retrieved from https://openreview.net/forum?id=S1eZYeHFDS

[77] Haokun Li, Bican Xia, Huiying Zhang, and Tao Zheng. 2023. Choosing better variable orderings for cylindrical algebraic decomposition via exploiting chordal structure. *Journal of Symbolic Computation* 116 (2023), 324–344. DOI: https://doi.org/10.1016/j.jsc.2022.10.009

[78] Christoph Lüders, Thomas Sturm, and Ovidiu Radulescu. 2022. ODEbase: A repository of ODE systems for systems biology. *Bioinformatics Advances* 2, 1 (2022), 3 pages. DOI: https://doi.org/10.1093/bioadv/vbac027

[79] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*. Curran Associates Inc., 4768–4777. Retrieved from https://dl.acm.org/doi/10.5555/3295222.3295230

[80] Markus Löschenbrand. 2020. Finding multiple Nash equilibria via machine learning-supported Gröbner bases. *European Journal of Operational Research* 284, 3 (2020), 1178–1189. DOI: https://doi.org/10.1016/j.ejor.2020.01.041

[81] Mohamed Malhou, Ludovic Perret, and Kristin Lauter. 2026. HATSolver: Learning Gröbner Bases with Hierarchical Attention Transformers. In *Proceedings of the 14th International Conference on Learning Representations*. 21 pages. Retrieved from https://openreview.net/forum?id=5C3LljOEGC

[82] Markos Maniatis, Andreas von Manteuffel, and Otto Nachtmann. 2007. Determining the global minimum of Higgs potentials via Groebner bases-applied to the NMSSM. *The European Physical Journal C* 49, 4 (2007), 1067–1076. DOI: https://doi.org/10.1140/epjc/s10052-006-0186-2

[83] Ernst W. Mayr and Albert R. Meyer. 1982. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics* 46, 3 (1982), 305–329. DOI: https://doi.org/10.1016/0001-8708(82)90048-2

[84] Jelena Mojsilović, Dylan Peifer, and Sonja Petrović. 2023. Learning a performance metric of buchberger's algorithm. *Involve, a Journal of Mathematics* 16, 2 (2023), 227–248. DOI: https://doi.org/10.2140/involve.2023.16.227

[85] Jasper Nalbach, Erika Abraham, Philippe Specht, Christopher W. Brown, James H. Davenport, , and M. England. 2024. Levelwise construction of a single cylindrical algebraic cell. *Journal of Symbolic Computation* 123 (2024), 102288. DOI: https://doi.org/10.1016/j.jsc.2023.102288

[86] Arthur C. Norman and P.M.A. Moore. 1977. Implementing the new risch integration algorithm. In *Proceedings of the Fourth International Colloquium on Advanced Computing Methods in Theoretical Physics*. 99–110.

[87] Dylan Peifer, Michael Stillman, and Daniel Halpern-Leistner. 2020. Learning Selection Strategies in Buchberger's Algorithm. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 7575–7585. Retrieved from https://proceedings.mlr.press/v119/peifer20a.html

[88] Dylan James Peifer. 2020. *Reinforcement Learning in Buchberger's Algorithm*. Ph. D. Dissertation. Cornell University. DOI: https://doi.org/10.7298/4fpv-bn09

[89] Lynn Pickering, Tereso del Río Almajano, Matthew England, and Kelly Cohen. 2024. Explainable AI Insights for Symbolic Computation: A case study on selecting the variable ordering for cylindrical algebraic decomposition. *Journal of Symbolic Computation* 123 (2024), 102276. DOI: https://doi.org/10.1016/j.jsc.2023.102276

[90] Bartosz Piotrowski, Chad Brown, Josef Urban, and Cezary Kaliszyk. 2019. Can Neural Networks Learn Symbolic Rewriting?. In *Proc. Workshop on Learning and Reasoning with Graph-Structured Data (at ICML 2019)*. 4 pages. Retrieved from https://graphreason.github.io/papers/40.pdf

[91] Albert Rich, Patrick Scheibe, and Nasser M. Abbasi. 2018. Rule-based integration: An extensive system of symbolic integration rules. *Journal of Open Source Software* 3, 32 (2018), 1073. DOI: https://doi.org/10.21105/joss.01073

[92] Robert H. Risch. 1969. The problem of integration in finite terms. *Trans. A. M. S.* 139 (1969), 167–189. DOI: https://doi.org/10.1090/S0002-9947-1969-0237477-8

[93] Eugenio Roanes-Lozano. 2022. A Maple-based introductory visual guide to Gröbner bases. *Maple Transactions* 2, 1 (2022). DOI: https://doi.org/10.5206/mt.v2i1.14425

[94] Ben Ruijl, Jos Vermaseren, Aske Plaat, and Jaap van den Herik. 2014. Combining Simulated Annealing and Monte Carlo Tree Search for Expression Simplification. In *Proceedings of the 6th International Conference on Agents and*

Artificial Intelligence - Volume 1 (ESEO, Angers, Loire Valley, France) *(ICAART 2014)*. SCITEPRESS - Science and Technology Publications, Lda, Setubal, PRT, 724–731. DOI: https://doi.org/10.5220/0004925707240731

[95] Amir H. Sadeghimanesh and Elisenda Feliu. 2019. Gröbner bases of reaction networks with intermediate species. *Advances in Applied Mathematics* 107 (2019), 74–101. DOI: https://doi.org/10.1016/j.aam.2019.02.006

[96] Massimiliano Sala. 2009. Gröbner Bases, Coding, and Cryptography: A Guide to the State-of-Art. In *Gröbner Bases, Coding, and Cryptography*, Massimiliano Sala, Shojiro Sakata, Teo Mora, Carlo Traverso, and Ludovic Perret (Eds.). Springer Berlin Heidelberg, 1–8. DOI: https://doi.org/10.1007/978-3-540-93806-4_1

[97] Vaibhav Sharma, Abhinav Nagpal, and Muhammed Fatih Balin. 2023. SIRD: Symbolic Integration Rules Dataset. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*. Retrieved from https://openreview.net/forum?id=WWDsbsgyhS

[98] Matthew C. Simpson, Qing Yi, and Jugal Kalita. 2016. Automatic Algorithm Selection in Computational Software Using Machine Learning. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 355–360. DOI: https://doi.org/10.1109/ICMLA.2016.0064

[99] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 3319–3328. Retrieved from https://proceedings.mlr.press/v70/sundararajan17a/sundararajan17a.pdf

[100] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (2 ed.). MIT Press. Retrieved from https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

[101] Alberto Testolin. 2024. Can Neural Networks Do Arithmetic? A Survey on the Elementary Numerical Skills of State-of-the-Art Deep Learning Models. *Applied Sciences* 14, 2 (2024), 16 pages. DOI: https://doi.org/10.3390/app14020744

[102] Quoc-Nam Tran. 2000. A Fast Algorithm for Gröbner Basis Conversion and its Applications. *Journal of Symbolic Computation* 30, 4 (2000), 451–467. DOI: https://doi.org/10.1006/jsco.1999.0416

[103] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature* 625 (2024), 476–482. DOI: https://doi.org/10.1038/s41586-023-06747-5

[104] Jan Verschelde. 1999. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Trans. Math. Softw.* 25, 2 (1999), 251–276. DOI: https://doi.org/10.1145/317275.317286

[105] Max Vladymyrov, Johannes von Oswald, Nolan Andrew Miller, and Mark Sandler. 2024. Efficient Linear System Solver with Transformers. In *AI for Math Workshop @ ICML 2024*. 5 pages. Retrieved from https://openreview.net/forum?id=qc2adlhAWF

[106] Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu et al. 2026. A Survey on Large Language Models for Mathematical Reasoning. *ACM Comput. Surv.* 58, 8, Article 209 (Feb. 2026), 35 pages. DOI: https://doi.org/10.1145/3786333

[107] Zichao Wang, Andrew Lan, and Richard Baraniuk. 2021. Mathematical formula representation via tree embeddings. In *Proceedings of the Third International Workshop on Intelligent Textbooks (CEUR Workshop Proceedings, 2895)*, Sergey Sosnovsky, Peter Brusilovsky, Richard Baraniuk, and Andrew Lan (Eds.). 121–133. Retrieved from http://ceur-ws.org/Vol-2895/

[108] Sebastian Wankerl, Andrzej Dulny, Gerhard Götz, and Andreas Hotho. 2021. Learning Mathematical Relations Using Deep Tree Models. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 1681–1687. DOI: https://doi.org/10.1109/ICMLA52953.2021.00268

[109] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS 2022)* (New Orleans, LA, USA). Curran Associates Inc., Article 1800, 14 pages. Retrieved from https://openreview.net/forum?id=_VjQlMeSB_J

[110] Stephen Wolfram. 2023. WolframAlpha as the Way to Bring Computational Knowledge Superpowers to ChatGPT. Retrieved from https://writings.stephenwolfram.com/2023/01/wolframalpha-as-the-way-to-bring-computational-knowledge-superpowers-to-chatgpt/

[111] Wanting Xu, Lan Hu, Manolis C. Tsakiris, and Laurent Kneip. 2019. Online Stability Improvement of Gröbner Basis Solvers using Deep Learning. In *2019 International Conference on 3D Vision (3DV)*. 544–552. DOI: https://doi.org/10.1109/3DV.2019.00066

[112] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2021. Dive into Deep Learning. *CoRR* abs/2106.11342 (2021), 1185 pages. *arXiv preprint arXiv:2106.11342*

Appendices

A Appendix: Buchberger's Algorithm

Recall that we introduced Buchberger's Algorithm to compute a Gröbner basis at the start of Section 4.

Suppose that we have a set of polynomials

$$F = \{f_1, \dots, f_m\} \subset R[x_1, \dots, x_n]$$

where $R[x_1, \dots, x_n]$ is a ring of polynomials. A Gröbner basis G is computed as follows:

(1) **Initialise:** Set
$$G = F.$$
(2) **Compute S-Polynomials:**
For each pair $(g_i, g_j) \in G$, compute
$$S(g_i, g_j) = \frac{\text{lcm}(\text{LT}(g_i), \text{LT}(g_j))}{\text{LT}(g_i)} g_i - \frac{\text{lcm}(\text{LT}(g_i), \text{LT}(g_j))}{\text{LT}(g_j)} g_j, \tag{1}$$
where lcm stands for least common multiple and LT denotes the leading term.
(3) **Reduction of S-pairs:** If the S-pair $S(g_i, g_j)$ is irreducible to zero modulo G, it is added to G in its reduced form.
(4) **Repeat:** New elements keep being added by repeating the algorithm from Step 2 until no more S-polynomials produce non-zero reductions.
(5) **Output:** G, a Gröbner basis for F.

Buchberger's algorithm is visualised in Figure 4 for an example set of polynomials.

B Appendix: Summary Tables

Table 3. Summary of Studies on CAD and ML

Study	Focus Area	Key Findings	ML Method	XAI	Drawback
Huang et al. (2014) [58]	ML to choose from three human-designed heuristics for variable ordering.	ML choices better than any one heuristic.	SVM	No	Dataset limited to three variables; later found biased.
Huang et al. (2016) [57]	ML to check if preconditioning with GB benefits solving non-linear systems with CAD.	On 1200 synthetic problems (2–4 vars), ML predicts when GB is beneficial.	SVM	No	Random dataset may not represent real-world data.
England and Florescu (2019) [44]	Replaced heuristics in [58] with ML ordering directly.	Removes human entirely from ordering choice.	KNN, DT, MLP, SVM	No	ML ignores ranking of incorrect orderings.
Florescu et al. (2019) [47]	Generated new features for CAD variable ordering.	outperformed human heuristics and benefitted from added features.	KNN, DT, MLP, SVM	No	Biased dataset.
Florescu et al. (2020a) [49]	Open-source pipeline for CAD ordering.	Random datasets with stats similar to real-world inputs.	KNN, DT, MLP, SVM	No	Dataset realism not evaluated.
Florescu et al. (2020b) [48]	Cross-validation based on CAD runtime.	Sensitive to non-optimal orderings, improving reliability.	KNN, DT, MLP, SVM	No	Still framed as classification in training.
Chen et al. (2020) [28]	Tackled fixed-variable limitation.	Iterative greedy selection + ML for final choice.	Deep Neural Network	No	—
Brown and Daves (2020) [17]	Studied polynomial processing order in CAD.	Used repeated binary classification to form ranking.	Deep Neural Network	No	Limited by random dataset shape.
del Río et al. (2023) [36]	Uncovered bias in NLSAT dataset.	Data augmentation and balancing improved ML performance by 63%.	KNN, DT, MLP, SVM	No	Data augmentation helps, but not optimal yet.
del Río et al. (2024) [38]	Reframed ordering as regression problem.	KNN, SVM perform better under regression; RF, MLP, XGB worse.	KNN, DT, MLP, SVM, XGB	No	Still supervised, data dependent.
Jia et al. (2024) [63]	Applied RL and GNN for ordering.	RL allows adaptive, data-driven selection.	RL, GNN	No	Advantage source unclear (RL vs GNN); performance drops on real.
John et al. (2024) [66]	Tested generalizability of extracted features.	Validated across original, balanced, augmented, shuffled datasets.	LR, KNN, DT, XGB, NN, GNN	Yes	Still supervised, sensitive to unseen data.
Chen et al. (2024) [24]	Created large-scale dataset for CAD ordering.	First labelled dataset large enough for deep learning (20K + augmentation).	RF, MLP, Transformer	No	Data generated randomly; unclear representativeness.
Pickering et al. (2024) [89]	First XAI use (SHAP) in CAD ML.	Identified novel important features; XAI-informed heuristic performed strongly.	—	Post-hoc (SHAP)	Based on [47] features; unclear if ML can generate better.
Florescu and England (2024) [50]	Modelled Brown's heuristic as constrained NN.	Trained on random, tested on real; strong performance.	Neural Network	Ante-hoc (small NN)	—
Jing et al. (2026) [65]	Pre-training and fine-tuning framework using six proxy algebraic tasks to learn deep polynomial structures.	2× speedup on DQ-3; near-optimal (1.07× optimal) performance on real-world SMT benchmarks.	Transformer	No.	low accuracy on deep symbolic operations like resultants

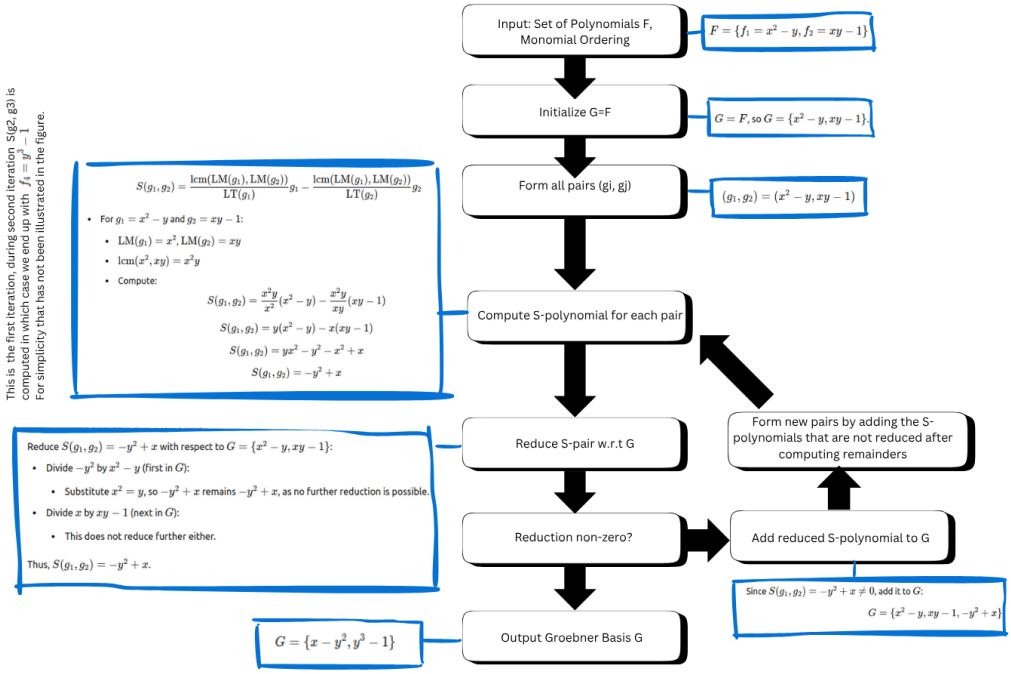

Fig. 4. A flowchart showing Buchberger's algorithm on an example.

Table 4. Summary of Studies on Gröbner Basis and ML

Study	Focus Area	Key Findings	ML Method	Drawback
FMN (2020) [80]	Approximate complex polynomial systems.	ML approximates systems to allow simpler Gröbner basis computation.	Evolution strategies	Potential loss of structure; not suitable for non-permutation-invariant systems.
Peifer et al. (2020) [87]	Optimising Buchberger's algorithm.	RL agent selects S-pairs for binomial systems, reducing computation.	Reinforcement Learning	Limited to binomials; trained and tested on random data.
Mojsilovic & Peifer (2021) [84]	Predicting polynomial additions.	Simple features perform better than complex algebraic ones.	LR, RNN	Indirect improvement only; focuses on prediction, not algorithmic enhancement.
Xu et al. (2024) [111]	Variable ordering for Gröbner basis.	Model selects variable order using coefficient sets.	Deep Learning	Overhead from ANN; applies only to permutation-invariant cases.
Kera et al. (2024) [70]	Gröbner basis produced directly by transformers.	Hybrid embedding scheme for reverse Gröbner basis generation.	Transformers	Needs large datasets; fails on out-of-range coefficients; only approximate, no easy validation.
Malhou et al. (2026) [81]	Gröbner basis computation using hierarchical attention transformers.	Tree-structured inductive bias matching polynomial hierarchy, combined with curriculum learning.	Hierarchical Attention Transformers	Still approximate with no easy validation; relies on curriculum learning to scale; performance beyond 13 variables unverified.

Table 5. Summary of Studies on Symbolic Integration and ML

Study	Focus Area	Key Findings	ML/RL Integration	Drawback
Lample and Charton (2020) [76]	Neural models for symbolic integration and differential equations.	Transformers solve problems that CASs cannot.	Seq2seq transformers	Limited evidence of generalizability; dataset quality concerns.
Albojwaid et al. (2023) [4]	Replication of [76] with modified data generation.	Larger training data, same methodology.	Seq2seq transformers	Little novelty compared with prior work.
Barket et al. (2023) [7]	Dataset generation with elementary functions.	Reverse-engineers Risch algorithm to address bias in [76].	Data generation for ML	Limited control over structure and length; restricted expression size.
Barket et al. (2024) [8]	Dataset generation for elementary and non-elementary functions.	Reverse-engineers Risch-Normal algorithm; adds variability and control.	Data generation for ML	Limited benchmarking of ML benefit; not extended to special functions.
Barket et al. (2024) [9]	Predicting optimal Maple sub-algorithms via ML.	Tree-LSTMs outperform standard LSTMs.	LSTM and Tree-LSTM	Does not generalise well enough to beat Maple on independent data.
Barket et al. (2024) [10]	Predicting applicability of Maple methods via ML.	Improves existing guards, adds new ones; XAI reveals missing simple guard rule.	Seq2seq transformers; LIGs	Most LIG outputs not interpretable; runtime impact of guards not evaluated.
Sharma et al. (2024) [97]	Selecting integration rules for functions.	Large dataset enables interpretable integration paths.	Step-level DNNs	High computation required for multi-step learning.
NNS (2022), MMP (2021) [40], [54]	University-level mathematics problems.	Frontier models can perform symbolic integration.	Yes	Not benchmarked for CAS use; outputs not validated.
Kubota et al. (2022) [72]	Critique of prefix notation use.	Modified [76] data; LSTMs outperform transformers via better token handling.	LSTMs	Not benchmarked for CAS use; outputs not validated.

Table 6. Summary of Studies on ML and Other Computer Algebra Tasks

Study	Description	ML Role	Limitation
1. Using ML to guide symbolic algorithms			
Berczi et al. (2023) [12]	Resolution of singular points as a game with RL agent.	Reinforcement Learning	Reward may be better tailored to problem.
Hashemi et al. (2024) [53]	ML chooses coordinate changes for putting ideals into quasi-stable position.	Feature engineered supervised learning	Tested only on random ideals; generalizability unclear.
2. ML algorithm selection for resultants			
Simpson et al. (2016) [98]	Predicts best resultant algorithm in Maple/Mathematica for given instance.	Feature engineered supervised learning	Trained platform-specific; cross-platform performance uncertain.
3. ML to improve CA for physics			
Alnuqaydan et al. (2023) [5]	Transformer replaces symbolic squared amplitude step.	Seq-to-seq Transformers	Downstream effect of errors unclear.
Kreuer et al. (2013; 2015) [73, 74]	Horner form optimization for CA system FORM.	Heuristic Search + MCTS	Tailored to physics; not general-purpose.
Ruijl et al. (2014) [94]	Horner form optimization for CA system FORM.	Simulated Annealing + MCTS	Specific to FORM; not generalised CA use.
4. Linear polynomial equation solving			
Dabelow & Ueda (2024) [33]	Symbolic stack solver heuristic transformation choices.	Reinforcement Learning	Does not use CA systems; scalability unclear.
Vaswani et al. (2024) [105]	Transformer trained directly to solve linear systems.	Deep Learning Transformer	Black-box; lacks step-by-step explainability.
5. Learning from the MATH-AI community			
Trinh et al. (2024) [103]	AlphaGeometry: solving Math Olympiad problems.	Hybrid RL + Deduction	Specific to geometry tasks.
Wang et al. (2021) [107]	Recognising formula equivalence.	Neural networks; tree embeddings	Dependent on quality of synthetic data.
Wankerl et al. (2021) [108]	Recognising equivalence, derivative, or variable substitution in polynomials.	Neural networks; tree embedding	Focus on relations and computation.
6. Explainable AI			
Charton (2022) [22]	Analyses transformer errors in linear algebra tasks.	Interpretability via Error Properties	No input-level explainability.
Charton (2024) [23]	Explains GCD prediction via structured experiments.	Experimental Explainability	Model internals still black-box.
Anthropic (2025) [6]	Reveals inner workings of LLM for integer arithmetic.	Mechanistic Interpretability	Does not give full interpretations.

Received 3 November 2025; revised 22 May 2026; accepted 12 July 2026