Symbolic Mathematical Computation 1965–1975: The View from a Half-Century Perspective

Robert M. Corless
Department of Computer Science,
Western University
London, Canada
rcorless@uwo.ca

Arthur C. Norman
Trinity College
Cambridge, UK
acn1@cam.ac.uk

Tomás Recio
Escuela Politécnica Superior,
Universidad Antonio de Nebrija
Madrid, Spain
trecio@nebrija.es

William J. Turkel
Department of History,
Western University
London, Canada
william.j.turkel@gmail.com

Stephen M. Watt
Cheriton School of Computer Science,
University of Waterloo
Waterloo, Canada
smwatt@uwaterloo.ca

Figure 1: Jean E. Sammet, the founding Chair of SIGSAM and General Chair and Program Chair of SYMSAC '66

Abstract

The 2025 ISSAC conference in Guanajuato, Mexico, marks the 50th event in this significant series, making it an ideal moment to reflect on the field's history. This paper reviews the formative years of symbolic computation up to 1975, fifty years ago.

By revisiting a period unfamiliar to most current participants, this survey aims to shed light on once-pressing issues that are now largely resolved and to highlight how some of today's challenges were recognized earlier than expected.

CCS Concepts

• **Computing methodologies → Symbolic and algebraic manipulation**.

ISSAC '25, Guanajuato, Mexico

ACM ISBN 979-8-4007-2075-8/2025/07
https://doi.org/10.1145/3747199.3747556

Keywords

History, Symbolic Computation, Computer Algebra

ACM Reference Format:
Robert M. Corless, Arthur C. Norman, Tomás Recio, William J. Turkel, and Stephen M. Watt. 2025. Symbolic Mathematical Computation 1965–1975: The View from a Half-Century Perspective. In *International Symposium on Symbolic and Algebraic Computation (ISSAC '25), July 28-August 1, 2025, Guanajuato, Mexico.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3747199.3747556

1 Introduction

The modern symbolic computation world is quite large in absolute terms, comprising on the order of a thousand active researchers and a million dedicated users, and even more people who use it occasionally. Symbolic computation has a very large impact through a variety of software packages and problem-solving environments.

The history of any important field is of intrinsic interest, and every researcher in it has an obligation to know the main currents of its history. Such knowledge also has instrumental value in that knowing the history can prevent new projects from repeating earlier mistakes and can help overcome similar obstacles. As the saying goes, 'Six months in the lab can save you three days in the library.'

Finally, the history of symbolic computation is a crucial component of the history of mathematical thought more broadly since it was one way that ideas of abstraction and axiomatic thinking reached much larger audiences from the mid-20th century onward [38, 90].

This paper tells part of the story of the history of symbolic computation in a way that addresses these desires. We have spent some time surveying, reading, and digesting the early literature on symbolic computation, and here we present some highlights of the period 1965–1975. However, this paper is not comprehensive. We welcome discussion, especially of things we have missed. If you have knowledge that has passed us by please get in touch so that extended and updated versions of this can be as complete as possible.

At the outset, we note that even now there is some uncertainty regarding the best name for the field most of us work in, even just in English! We will not be strict. ISSAC refers to the field as "Symbolic and Algebraic Computation" while ACA uses the term "Computer Algebra". JSC drops the word "Algebraic" and thus it is only "Symbolic Computation" where the main ACM subgroup SIGSAM went for "Symbolic and Algebraic Manipulation". We will use the terms largely interchangeably. We will use "Symbolic Computation" to mean symbolic mathematical computation, since other fields also use that term.

We do not attempt here to define what is, and what is not, a topic relevant to computer algebra. It is possible to disagree on whether a topic should be included or not. We could give historical examples of this discussion going right back to 1966.

At one end of the spectrum have been groups that have built software to be applied in a range of mathematical and engineering domains. At the other end have been those concerned with theory. To some extent, this can be seen as the continuation of earlier work to rebuild mathematics from a constructivist standpoint. The mix of tensions and cross-pollination across the different approaches is part of what gives the subject and its associated conferences some of their flavour.

We have organized our report chronologically in sections, ending in 1975 (a half century ago). Activity in each period had its own distinctive dynamic. Within each section, we use the device of classifying contributions as those for software systems, applications, algebra and formal proof. We may view these as contributions by the heirs of Turing, Laplace, Hilbert, and Tarski respectively. [1]. Of course people contributed in more than one way, and in reality these four groups are insufficient to truly cover the ground. We might also add the heirs of Abel to cover group theoretic contributions, the heirs of Noether to cover symmetry and invariants, the heirs of Fermat and Gauß to cover number theoretic contributions, or the heirs of Euclid for geometry.

Perhaps we could classify numerical computation as the work of the heirs of Cauchy; by and large we *exclude* such work here, except for multiple precision and except insofar as it was used for exact computation. See the impressive work [12] for an 800-page journey through just the history of numerical linear algebra, as a partial recompense for this exclusion.

Our focus is on symbolic mathematical computation for programmable computers. We highlight key algorithmic developments in this context.

We have chosen to frame our work by focusing on the contributions of Jean E. Sammet (1928–2017), who contributed signally to the early symbolic computation world [34] and took prime responsibility for the SYMSAC'66 conference that is being celebrated here as having grown to become ISSAC. Among her further achievements, she was the founder and first Chair of SIGSAM, the ACM Special Interest Group on Symbolic and Algebraic Manipulation. She was the lead designer of FORMAC, which was the first commercially successful programming language for computer algebra, and was instrumental in the development of COBOL. She later became the first female president of the ACM itself.

Sammet was also a historian of computing. "From childhood on, I hated to throw papers away. As I became an adult, this characteristic merged with my interest in computing history. As a result, I created important files and documents of my own, and became concerned with having other people publish material on their important work so the facts (rather than the myths) would be known publicly." [44] She wrote an encyclopedic book [83] on the early history of programming languages (concentrating on those invented in the USA), including those for computer algebra, and later wrote a condensed paper on the same subject, together with a look to the future [84]. As well as her having started the SYMSAC conference series and founding key early groupings and publications, her output has helped us to see how those active in the field saw it at the time, and in particular what they viewed as part of the subject: so what we report here is less tainted by our modern views than might otherwise have been the case.

2 1964 and before

By the 1930s mathematicians had developed a concept of an "effective procedure" for completing a task, even though computers as we know them were not available. The 1926 paper of Grete Hermann [59], translated to English in 1998 [60], was cited in [20] (the table of contents of which is in [19]) as being of special importance. These century-old works really set up the idea of what could be done algorithmically, and the biggest result associated with it was Gödel's Incompleteness Theorem which effectively dashed the hopes of those seeking to rebuild all of mathematics on constructive foundations. A key point to make here is that in deriving that result, Gödel publicised the fact that any formula could be encoded using a number.

The Turing machine was defined in the same decade. It established a framework for mechanical computation that could handle mathematical equations and a desire to solve them exactly or prove properties of them. It also suggested using one computational model to simulate another: in effect introducing the concept of an "interpreter". In the same time-frame, the American logician Alonzo Church introduced the lambda calculus as a model of computation. That the lambda calculus and Turing machines provide equivalent computational power is the Church-Turing thesis. Church also

[1] The authors of [73] label the parts of their book with Euclid (for the Euclidean algorithm and algorithms in general), Newton (for Newton's method but also for some applications), Gauß (for finite fields), Fermat (for number theory and cryptography), and Hilbert (for algebra, in a similar way to how we use it). Notice that the heirs of Turing are not present in that book. We suspect that this was so that the book would age more slowly.

proved that problem of deciding validity of formulas in first-order logic is unsolvable [22, 23], which has implications throughout computer algebra, including subsequent work on the decision problem [35] and later Richardson's theorem on the undecidability of symbolic real expressions [76]. Of course, all of that work was theoretical and much of it did not see general algebraic transformations and simplifications as the main objective, but it provided a basis for those who came later.

By the late 1940s and through the first half of the 1950s an increasing number of mathematicians became able to access electronic computers. Several of them had prior experience using electromechanical calculators to compile tables of special functions or to predict the trajectories of shells: work that was numerical in style. But some looked at problems that might be described as pure rather than applied mathematics. Perhaps the main focus was on support for logic, inference, and proof checking. Today these are still official topics of ISSAC, but in the early years schemes set up to handle them became directly applicable to symbolic elementary algebra.

A notable contribution was IPL (Information Processing Language) by Allan Newell, John Clifford Shaw and Herbert A. Simon[2] (c. 1956) which introduced list processing, recursion and dynamic memory allocation. IPL further featured functions as arguments and a form of multi-tasking: a major task it undertook was to show that the proofs in *Principia Mathematica* could be checked mechanically. It was an early example of a language or system that ended up available on a range of different computers. Newell and Simon demonstrated their automated reasoning program at the 1956 Dartmouth Summer Research Project on Artificial Intelligence, which was co-organized by John McCarthy.

FORTRAN made its first appearance in an IBM memo dated November 10, 1954, and FORTRAN II was released in June 1958. See [83] for a detailed discussion. As will be seen in the next section, FORTRAN later became an important base for mechanized symbolic computation. But from its inception (indeed from the time of Autocode some years earlier) the issue of parsing algebraic formulae had been addressed. From technical reports published even some years later it is clear that this was a challenging task for quite some time. It is hard to have an algebra system without a way for the system to accept reasonably natural mathematical input! Indeed some of the very earliest attempts used what now seem like bizarre tabular notations for both problem presentation and delivery of results.

ALGOL was developed in the period 1955–1957 by work started by a committee struck by GAMM[3] [83, p. 173]. COBOL was developed extremely rapidly, taking just the last six months of 1959 [83, p. 331]. LISP, very much a successor to IPL, originated at around the same time (the key publication [68] was in 1960), and those working close to McCarthy started to use it for a range of symbolic calculations.

As with IPL, some of the very earliest activity concentrated on logic rather than algebra. Efficient list processing, the first LISP compiler (1962), extended precision integers and floating point values (including elementary function evaluation), and garbage collection would eventually build on this to provide necessary infrastructure for symbolic algebra.

Somewhat later, the first in the SNOBOL family of string processing programming languages was initiated [42, 43] at Bell Telephone Laboratories. This emerged from the work on SCL [53, 64], "a Language for Symbolic Communication," used for symbolic integration, factoring of multivariate polynomials and the analysis of Markov Chains [53]. Symbolic computation has continued to be one of the intended application areas of the SNOBOL languages. For a summary of the early history of programming languages, see [84], which has a beautiful map showing lines of descent.

The state of the art of symbolic computation software in the period covered by this section has been captured in Sammet's survey article [81]. On the algorithmic side, of particular note is the emergent interest in faster methods for multiplication by Karatsuba [61] and leading to Toom-Cook multiplication [28, 93].

The situation at the end of the 1950s can be looked at in two quite contrasting ways. Almost nothing that we would now view unambiguously as an ISSAC topic had yet been published. But several projects had begun and a very substantial body of fundamental work would lead to tremendous growth the following decades. From that perspective the bounty of the 1950s included:

(1) Computers of increasing power and reliability became fairly broadly accessible to people wishing to develop advanced applications;
(2) High level languages rather than machine code became a practical strategy for developers, with some hope for cross-platform portability. Fortran and LISP were the ones that were widely used then and have lasted, but ALGOL existed and has influenced almost everything since, and at the time string processing such as SNOBOL were seen as core technology for symbol manipulation;
(3) List and tree data structures with automatic storage management was an understood concept – either built into the language used or implemented as part of the program being written;
(4) There were links between practical and theoretical work and workers relating to algorithms and computation. This spanned cost analysis of algorithms where today the results would be reported in big-O notation, formal models of computation such as the lambda calculus and forged bridges between theoretical development and practical problem-solving applications;
(5) The body of experience in proof checking and calculations involving logic was such that adapting it to work with general classical algebraic formulae could seem natural;
(6) Initial planning or work was under way on many significant projects that will be described in the next section.
(7) The concept of artificial intelligence inspired much work on carrying out procedures that humans find challenging, and working with algebraic formulae certainly fell into that category.

None of this work was isolated from wider social and cultural concerns. RAND Corporation, where IPL was developed, had spun off from a post-war US Air Force project to plan for future weapons. We view a verification of Russell and Whitehead as perhaps an

[2] See the description of the contributions of Newell, Shaw, and Simon at the History of Information website. https://www.historyofinformation.com/detail.php?id=742
[3] https://www.gamm.org/

unusual future weapon and celebrate the support that project was given. The Soviet launch of Sputnik in October 1957 led President Eisenhower to establish the Advanced Research Projects Agency (ARPA) early the following year. One of its missions was to give the US the capacity to launch and use spacecraft. This called for work on automating calculations in celestial mechanics and orbital dynamics where pure numeric computation became strained and symbolic work was the best option. ARPA would go on to fund many of the projects we describe below.

3 1965–1970

At the instigation of Jean E. Sammet, the ACM Special Interest *Committee* on Symbolic and Algebraic Manipulation (SICSAM) was formed in 1965 by George Forsythe, then President of the Association for Computing Machinery. In a 2006 oral history [8], Sammet recalled Forsythe writing back to her saying "I think that's a fine idea. You are now the Chairman..." Years later, Sammet recalled that one of the main motivations to form SICSAM was to have a forum where staff from different companies could share ideas [85]. Later this "Committee" became a "Group" and ACM SIGSAM was born. The following year, Sammet organized the first SYMSAC, which took place in Washington, DC and eventually led to the ISSAC conference series. "Everybody that went thought it was wonderful, and when you expected them to be out at the bar, they were in sessions!" she also recalled.

Although the ACM Digital Library does not record who was the General Chair of that conference, Sammet said she served as General Chair, Program Chair, and Chair of SICSAM, "a troika of one." Some of the papers were published in the *Communications of the ACM* after the conference. They include works on systems, on applications, on techniques for differentiation, and something that at first glance *looks* like it's for teaching—it's entitled "Grad Assistant"—but is not; the idea was that the program would replace your graduate student research assistant and do your drudge computations for you!

There is another quotation from Sammet's interview which is quite sobering in view of recent developments in the relationship between SIGSAM and ACM, and indeed between some journal editorial boards and their publishers.

> One of the interesting things involving the SIGs and SICs occurred when some of the leaders of a SIG or SIC became annoyed with ACM. Periodically, a Special Interest Group or Committee Chair would get very annoyed with ACM and say, "We're just going to leave ACM. We don't want anything more to do with you; there is too much bureaucracy. We're going to leave." And I would sit down and talk to this person and I would say, "What does "leave" mean?" "Well, we're going to go form our own organization." And I said, "Well what about the money?" "Oh, well we've got lots of money in our treasury." I would then say: "You may not know it, but the ACM bylaws say that all the money in a Special Interest Committee or Special Interest Group belongs to ACM. The ACM bylaws clearly say that if a Special Interest Group or committee dissolves, the money just goes to the general ACM funds. You may leave ACM. You may take your key people with you. That does not dissolve the Special Interest Group or Special Interest Committee. There will surely be other people who are willing to run it. Maybe they're not as good as you are, but there will be people who are willing to run it, and they will have the money." And so he would say, "Oh." And he would go away and think about it, and that was the last that I would ever hear of this resigning from ACM, because it was a pretty meaningless gesture.

We see therefore that Sammet had a significant role in the creation of SIGSAM and indeed in the reconstruction of ACM itself at that time.

One more quotation, this one with an ironic outcome. Sammet said:

> There was a very strong Special Interest Group on numerical analysis [SIGNUM] in ACM, and at some point I think I contacted them, and, in a presumably polite way, they told me to go away and not bother them. They were interested in numerical analysis, and they didn't want any of this non-numerical stuff floating around. And that attitude lasted for a very, very long time.

When SIGNUM closed up shop around the turn of the 21st century, during the tenure of one of the present authors as Chair of SIGSAM or shortly thereafter, its remaining assets were assigned to SIGSAM. So the present ACM SIGSAM is the heir to both groups.

We mention one more item of interest from the archives of the SICSAM Bulletin (later the SIGSAM Bulletin, later *Communications in Computer Algebra*). The first Editor of the Bulletin was Peter Wegner[4]. He passed the torch after a year to John Young, who announced in his first message to the readership[5], that there would be a European sub-editor to whom correspondence should be addressed from that side of the Atlantic. That sub-editor was Professor Sir Maurice Vincent Wilkes, of Cambridge University. Wilkes won the Turing Award the next year, in 1967, and was knighted in 2000 for his many contributions to computer science. Early issues of the Bulletin also included a note on curriculum design for computer science by Donald E. Knuth (ACM Turing award 1974) co-authored with Peter Wegner. We will also note that "Tini" Veltman, whose 1963 algebra system Schoonschip [95] is another candidate for being "first," went on to become a Nobel Laureate. It seems that Sammet was right: there was significant interest in the community in symbolic and algebraic manipulation, and that attracted participants of extraordinary calibre.

Sammet identified the language ALGY as the first published step towards a general-purpose language for computer algebra [10]. The authors were Myrna D. Bernick, E. D. Callender, and J. R. Sanford. The system ALGY seemed to allow basic polynomial manipulation. Sammet credited this work with ideas that led to the creation of FORMAC in this period. FORMAC has been called the "first commercially successful computer algebra system." Elaine R. Bond's history of FORMAC, published in the *Proceedings of SYMSAC '66* acknowledged the influence of "other similar systems such as ALPAK,

[4] Peter Wegner (1932–2017) was a professor of Computer Science at Brown University from 1969 to 1999. His work on object-oriented programming is considered seminal.
[5] On p. 1 of Issue #5

ALTRAN, and Formula ALGOL" and cited a paper of Sammet's that appeared in technical report form in 1965, and later in [82].

ALPAK (Algebra Package) was a set of SNOBOL routines and macros for symbolic manipulation of very large rational algebraic expressions. It was implemented by W. Stanley Brown and colleagues at Bell Labs in the early 1960s. This was succeeded by ALTRAN, "a highly portable implementation of both algorithms and compiler... [which] included algorithmic advances in the handling of multivariate polynomials and macro generation to tailor FORTRAN code to make full use of the characteristics of particular hardware. Its features included a run-time environment with dynamic storage allocation, recursion, symbolic dumping, and error handling". One challenge raised by the ALTRAN team was finding multivariate greatest common divisors. This led to intensive work in the early 1970s before being satisfactorily resolved [69].

It's part of the folklore of the symbolic computing community that 'the first program for symbolic differentiation was written before FORTRAN existed,' and this turns out to be true. In [7], a paper describing a system "MANIP" that could differentiate, we find citations that go back to 1962. That 1962 paper [55] cites two 1953 papers, one by Kharimanian and one by Nolan, and describes the process by which the programs had to be used: in each case the expressions had to be translated into a compact code before the programs could be run. These papers are credited in [83] as being the first.

Drawing further from [55], we infer several things. First, these early differentiation programs were seriously limited by the capacity of the computers used and a consequence was that the input and output of expressions had to be in really ugly restricted formats. The one in [55] accepted input in forms that looked like

```
A * X P 2 -I- B * X -4- SIN.(C * X)
```

They used "P" where today we might use "^". They parsed the input into a kind of Łukasiewicz prefix form [67] in a table and thought of this more as a table than as a tree.

The applications that they mention include a guided missile system, and they say "Equations consisting of three hundred or more symbols have been successfully differentiated with resultant derivatives of length in excess of seven hundred symbols. The equations under consideration involve six independent variables and four dependent variables."

If one can differentiate, then one can compute Taylor series. It turns out that for efficiency one should be careful, because naive methods lead to combinatorial growth in the length of symbolic expressions, as soon became painfully clear. So-called *automatic* differentiation emerged later to address this, but early work included [75]. That report described a program that generated FORTRAN and machine code for the particular computer in use at the author's institution in a way that we might classify as automatic differentiation today.

The papers so far highlighted here are concerned with system construction. Papers from applications also appeared in SYMSAC '66, such as [30], which described a FORMAC program to solve linear initial and boundary-value problems for ODE. The author, Elizabeth Cuthill (1923–2011), is perhaps most famous for her work on the *Cuthill–McKee* algorithm, which permutes sparse matrices in a way to reduce bandwidth. The paper describing that work is [31]. It's a judgement call to say that that work was "symbolic computation," but since it is so famous and so useful and has been cited thousands of times we are quite motivated to claim it for the community!

Another contribution in the applications line is [32] which described manipulation of Poisson series, that is, series where the terms are of the form

$$x_1^{k_1} x_2^{k_2} \cdots x_m^{k_m} e^{y_1} e^{y_2} \cdots e^{y_n} . \tag{1}$$

Such series[6] were, and are, in great demand for solving problems in celestial mechanics; see the famous paper [37].

James R. Slagle (1934-2023), who completed his 1961 dissertation under Marvin Minsky [88], described a LISP program called SAINT which could find antiderivatives of many functions. Slagle, who was blind, was also a chess champion. He won the first Over-the-Board tournament of the US Braille Chess Association.[7] Joel Moses (1941–2022), who began his doctorate in 1963, redesigned Slagle's SAINT into a symbolic integration program called SIN completed in 1967. It was also written in LISP and advanced the use of knowledge based systems rather than tree search for AI [70]. This work in LISP coincided with the creation of Project MAC at MIT (1963). Its first director was Robert M. Fano. Out of this project the MACSYMA system would be born.

REDUCE was also created in the early-mid 1960s [57, 58]. The first published *use* of REDUCE was [56], where it was reported that six months of human labour had been reduced to fifteen minutes of computer time. REDUCE 2 appeared in 1970. One of the present authors has the full working source of that program, and its manual, from which we find that REDUCE 2 did not have arbitrary-precision integers or bigfloats, and instead relied on whatever floating-point system was supplied by the underlying machine. Given the state of floating-point arithmetic in the years before the IEEE Standards, this seems natural. REDUCE had to wait until 1979 for a bigfloat package [86]. COBOL, in contrast, already had facilities for specifying the number of figures before and after the decimal point fairly freely, which was another milestone for Jean Sammet. The first (and probably only) algebra system implemented in COBOL was described in 1976 [46]. To be fair, that system, while slow, actually had some quite interesting features.

MATHLAB was also a product of the mid-1960s, created at MITRE[8] by Carl Engelman (1929-1983) [41]. One of his rules stated "MATHLAB is intended for the physicist, not the programmer.", while another insisted:

> The computer, as viewed by the user, must be intimate and immediate. The user should have next to [his] desk a console consisting of a typewriter or, preferably, a typewriter and a scope. Economy might, in some cases, dictate the substitution of a plotter for the scope. These are connected to a large, fast, on-line, time-shared digital computer. [He] communicates with that computer by typing messages on his typewriter or

[6] Or the more or less equivalent ones with sines and cosines in place of the complex exponentials

[7] https://web.archive.org/web/20160502160843/http://www.americanblindchess.org/potb.htm

[8] MITRE was a military think tank that was spun off from MIT Lincoln Labs in 1958. Its first employees were developers of the SAGE system which provided air defense during the Cold War.

> by means of a light-pen on the scope. The computer replies by means of the same machines. It types both messages and equations. On the scope it displays both equations and graphs. Above all, the response time to the user's requests must be short.

It would be at least twenty to thirty years before that goal would be widely achieved. Mathlab 2 introduced two-dimensional output (subscripts and superscripts in ASCII format).

SCRATCHPAD I, written in LISP by James Griesmer, Dick Jenks and later David Yun [52], started in 1965 but was never publicly released. First versions are important for history, though: Griesmer's knowledge was later incorporated into SCRATCHPAD II by Dick Jenks and his team.

Now, we must include a seminal contribution in the area of algebraic algorithms, namely the 1965 PhD thesis of Bruno Buchberger [14] which introduced Gröbner bases to computer algebra, and which was later translated to English in [18]. The thesis was first followed by [15], which itself was translated to English in [17]. These major works and the Buchberger algorithm did not make their way into computer algebra systems until much later; in fact, not until after 1976 and the publication of [16]. Nevertheless these were important milestones for the era.

Before Gröbner bases, though, there was elimination theory using resultants. Significant prior work includes [54]. One very important paper of the late 1960s, now less well known than it deserves to be, is [63]. It made the claim that, for multivariate polynomials, Bézout matrices are superior to subresultants both in respect to speed and in respect to numerical stability. In view of the very pessimistic recent result [72] this may merit a second look with a modern lens.

Other important papers in polynomial arithmetic include Berlekamp's factorization of polynomials over finite fields [9] and Lipson's use of Chinese remaindering [66] for homomorphic methods.

A fundamental and practical advance was made in [1], where an efficient method for solving linear systems of equations over the integers was given. This was the first of the "fraction-free" methods. Later, in [2], this was extended to general integral domains.

A huge development in the theory of integration in finite terms was published in this time period, namely the papers of Robert H. Risch giving (in outline) an algorithm to either find an antiderivative of a given elementary function, or to prove that no such expression was possible. See [77], [78], and [80] for a more accessible introduction. See also [79], which seems to have been the paper which converted the earlier analytic techniques of Liouville and of Ritt into the algebraic terms that we know now. Implementation of this algorithm took quite some time and effort; the *Proceedings of EUROSAM '79* contain four important papers by Norman, Davenport, Trager, Moses and Zippel which helped to bring the development to a more satisfactory state. Of course, research continues today.

At the risk of venturing too far outside of the symbolic computation world, we mention the pioneering work of Stephen A. Cook on the complexity of computation of functions, beginning with multiplication, for example the 1969 paper [29]. This was soon followed by fast multiplication by Schönhage and Strassen [87]. The paper by Strassen [91] was a fundamental advance in improved complexity for linear algebra. The field of "computational complexity" has had a profound effect on research in computer algebra ever since. See, for instance, the textbook [73].

We now mention the 1969 undecidability result of Richardson [76], and the 1970 paper of Caviness [21]. According to Joel Moses [70], Richardson's result was a bit controversial at the time because of his inclusion of "absolute value" in the collection of functions allowed in the expression to be processed. Nonetheless, the fact that recognizing zero is undecidable over such a simple class of expressions remains remarkable. Specifically, if the expression E contains $\ln 2$, π, $\exp(x)$, and $\sin(x)$, then the problem of determining if $E < 0$ is undecidable. Adding the function $|x|$ to the list of possibilities makes determining if $E = 0$ undecidable. As discussed in [21] this result has many implications for practical symbolic computation, because simplification is fundamental. Because (some) simplification problems are undecideable, we must sometimes rely on heuristics, and this leads to a forest of difficulties even today.

4 1971–1975

Computer algebra systems become truly international in this period. The program REDUCE 2 had been widely distributed and groups all over the world started to contribute.

The very efficient and compact CAMAL (for Cambridge Algebra Language), by David Barton, Stephen Bourne[9], John Fitch and others, started its gestation in about 1968 according to the delightful little history [45] (see also [3, 4]) but was not published until 1971.

MACSYMA became publicly available in 1971, after developments mentioned in the previous section. The history of MACSYMA is sketched broadly in [38], from the point of view of a historian of computing. The comments there might be fascinating to the insider: we see that in addition to providing tools for human use, MACSYMA forced the humans to adapt to the tools. HAKMEM (a February 1972 AI Lab memo) also came out of MIT around this time. Compiled by Michael Beeler, R. William Gosper, and Richard C. Schroeppel, it described 'little known data' of interest to computer hackers, 'to save some duplication of effort–except for fun' [6]. It has become known to subsequent generations through references like [65] and [98]. MACSYMA was weighted more heavily towards the knowledge of special functions than were the other systems at the time, and this memo reflects an important emphasis of that software system.

ALTRAN continued its development during this time. We note that Morven Gentleman (1942–2018) is credited in the fourth edition of the ALTRAN manual as being one of the developers, probably in the early 1970s; some of us knew him later as an advisor for the creation of the 1980s program Maple. Gentleman published a paper in the SIGSAM Bulletin using ALTRAN for Truncated Power Series [49]. Another related result was Gentleman showing that, unexpectedly at the time, the optimal multiplication chain for powering a polynomial was repeated multiplication by the original polynomial [48].

Also in this period, George Collins (1928–2017) introduced his SAC-1 system for polynomial arithmetic [24]. This system, written in FORTRAN, included functions for polynomial GCD, factorization,

[9] Stephen R. Bourne is known for the Bourne shell, for work on ALGOL68, and for having been President of the ACM. See his Wikipedia page https://en.wikipedia.org/wiki/Stephen_R._Bourne.

resultants, exact real zero calculation, partial fraction decomposition, rational function integration, and solution of systems of linear equations with polynomial coefficients.

It is hard to overestimate the influence of SCRATCHPAD II project that followed SCRATCHPAD I in the 1980s. The heady "pure research" environment of IBM T.J. Watson Research Center provided a hub for the exchange of ideas and technology. The interchanges between visiting researchers and all kinds of scientists and engineers led to many downstream effects. The principal feature that emerged from SCRATCHPAD II and that differentiated it from other computer algebra systems (before or since) was its formalization of the domains of computation. This enabled both rigour and efficiency in a way that was both challenging (to the user, sometimes) and satisfying to the researcher.

An example paper from this time is [74]. This paper appeared in ACM *Transactions on Mathematical Software*, which even then was an excellent vehicle for publication[10]. A comparison to the report [75] published ten years earlier shows considerable development: we see for instance what is now called "lazy evaluation," as well as considerably improved ease of use. Portability, however, remained an issue for both works, although for completely different reasons.

Leaving systems for the moment and considering algorithms, major progress was made in this time period on the important problems of multivariate GCD [71] and factorization [96, 97] using ideal-adic methods [100]. This work—building on the work of others as it does—marks the take-off of sophisticated modular methods in computer algebra systems, which remain crucial system tools for efficiency.

Several people have since observed that, *in applications*, multivariate polynomials factor frequently and usefully. This implies that the results of these fundamental papers have had, in all likelihood, an outstanding if underappreciated effect on the utility of symbolic computation. Since multivariate polynomials with coefficients "chosen at random" mostly do not factor at all, perhaps the extreme utility of this work could not have been anticipated.

Perhaps the most highly-cited work ever published in the SIGSAM Bulletin was George E. Collins' follow-up to [25], namely the abstract [26] which is just outside our time frame, although the first one is inside our time frame and is the one that has the actual results (the second is just an abstract[11]). As of January 2025 that abstract had been cited more than two and a half thousand times. This reflects the importance of the topics of cylindrical algebraic decomposition and quantifier elimination. Collins cites Tarski [92] in that first-mentioned paper for the foundational work.

Descriptions of other applications of computer algebra continued to be published in this time period. For instance, consider [5], a very substantial paper of over seventy pages (including a seven-page bibliography). This paper surveyed various applications in physics: celestial mechanics, general relativity, and quantum electrodynamics. It also gave an overview of computer algebra systems for physicists (not just REDUCE), and from this paper we learn that SIN by Joel Moses (mentioned above) already used Hermite reduction to perform integration by partial fractions.

5 Towards more modern times

We leave off discussion of the history of symbolic computation with a few short remarks leading to the present day. We do not wish to give the impression that we believe that the current state is uniformly better than that of the past. Indeed we do *not* hold that opinion, and instead lament some lost opportunities (for efficiency, for example). But there is no question that the fifty years since 1975 have wrought great change and in many cases very significant advances. We do not tell that story here even in the laconic style we used for the years 1965–1975, but rather just mention a few items.

- The first is the 1982 book edited by Buchberger, Collins, Loos, and Albrecht. The table of contents is printed in [19], where we see that a comprehensive overview of the field is attempted for the first time. A significant collection of historical material is present in the volume itself. This clearly marks a degree of maturity in the field;
- The first textbooks on computer algebra (it may be invidious to mention any particular one, but for instance the book [33] was translated to many languages);
- Wen-tsün Wu[12] and the method of characteristic sets 1978 [47], starting the topic of automated deduction in geometry by computer algebra tools that is still quite active (ADG conference series, see https://adg2023.matf.bg.ac.rs);
- The founding of the *Journal of Symbolic Computation* in 1985;
- The founding of the Research Institute for Symbolic Computation (RISC)—Linz in 1987, of the Key Laboratory of Mathematics-Mechanization (KLMM)-Beijing, also in 1987, and of the Ontario Research Centre for Computer Algebra (ORCCA)-London, Ontario, in 1997;
- The rise of the most popular modern commercial systems, namely Maple and Mathematica, and the MATLAB Symbolic Toolbox.
- The creation and development of Cayley, later MAGMA, Macaulay, CoCoA and Singular. These last three were all started in the 1980s and specialized in computational algebraic geometry;
- The D5 principle, as expounded by Dominique Duval and others [36, 39];
- The merging of the SYMSAC conference series (held every 5 years from 1966 to 1986) with the interleaved European EUROSAM/EUROCAM/EUROCAL stream to form the named ISSAC stream in 1988, together annual since 1981 (for conference chairs and program chairs see [27]);
- Developing visions of the future [11, 99];

[10] Some of the present authors feel that this journal was not as well-used historically by the symbolic computation community as it might have been, perhaps because it was viewed as primarily a "numerical" software journal. This (very symbolic) paper shows that that perception was incorrect. There were a few further symbolic computation papers in TOMS since this paper, and they tended to have good impact outside our community, but there were not that many. More might have been better.

[11] At times one gets the idea that citation culture is irrational. We use Google Scholar to count citations because of its wide disciplinary coverage. Although flawed, Google Scholar citation counts are just a proxy for impact anyway, and because we use this consistently at least it's a reasonably fair comparator. Being off by, say, ten percent is entirely likely, but with this many citations that doesn't matter.

[12] In the Western manner of family name last, his name is spelled variously Wenjun Wu, Wen-Tsun Wu, Wen-tsun Wu, and Wen-tsün Wu. He himself used the latter, and so we do the same.

- Macsyma, AXIOM and REDUCE, all of which had previously been either commercial or otherwise subject to redistribution constraints were re-licensed as open source and the general purpose capabilities they offered thus became available to all cost-free. This also means that the algorithms and techniques embedded within them could be studied by anybody who was interested, and various volunteers support them right up to the current day. These will now count as among the oldest software products still being maintained and used;
- Continuing development of applications (for instance, the Canadarm[13])
- Continuing development of practical methods for exact solution of linear systems [40];
- Portmanteau mathematical, engineering and educational support software, led by SAGEMATH where symbolic computation is supported alongside numerical work, visualization and document preparation and presentation;
- Hardware design motivated by the special needs of our subject, including both the Xerox D-series machines, the MIT Lisp machines[14] [51] and FLATS[50]. Some of these hardware projects have left a lasting impact on modern ways of using personal computers, while others were eclipsed by the rising tide of cheap fast processors.

5.1 Other histories

We have mentioned already some historically-oriented papers. Then there is the telegraphic list at Brian Evans's History of Computer Algebra[15].

In [94] we find a history of computer algebra starting in the 1980s, traced back through the Spanish engineer and inventor Torres–Quevedo (1852–1936) and before. That paper takes as its actual starting point the results of SYMSAC '86 and the meeting of the one of the present authors there with Wen-tsün Wu, and works its way backward.

6 Significant omissions

We have not discussed the history of the use of computer algebra in education, even though it has been very important for many members of our community since the beginning. We might have included the creation of the computer language LOGO in 1967. Instead we point to the history at [89], and leave that story for another day.

Automatic reasoning, then called theorem proving, was mentioned as an interest of the members of SICSAM in the very first issue of the Bulletin. Yet we have not mentioned much about the developments in 1965–1975 apart from the quantifier elimination work. There is much more to say.

We mentioned the two-dimensional math output of MathLab, but the whole subject of human-computer interaction has had a "test laboratory" in computer algebra systems. Input of complex objects, control of highly structured objects where transformations from one class to another are routinely desired, and sophisticated visualization of the output have all been desiderata. Simultaneously there has been a strong need for standardized dissemination of the results.

Neither have we talked about visualization or illustration of mathematical concepts. One very amusing visualization in the 1975 Bulletin is by Donald E. Knuth [62], where in describing the then-current state of his "Volume 2, Seminumerical algorithms" he includes a hand-drawn chart of editing changes.

7 Concluding remarks

We have given a brief summary of work in computer algebra and symbolic computation, concentrating on the period 1965–1975, ending fifty years ago. This time period was chosen in part because this ISSAC is the fiftieth since SYMSAC '66, the founding conference organized by Jean Sammet. We have spent a bit more space in this paper in detailing her contributions, which we believe were formative for the community.

During 1965–1975, several important features of our current set of tools were established: efficient methods for dealing with polynomials and series, effective methods for differentiation and integration, methods for solving linear and nonlinear algebraic equations, and manipulations of special functions. One major idea driving our subject early on was Artificial Intelligence. Another could loosely be termed "solving equations". In each case the work was notable for being sharply constrained by the amount of memory that was available.

A half-century and more later, we can see the discipline of "symbolic computation," which had split off into its own island state for a while, beginning to move back to merge with other sorts of computer work, bringing with it its style of mathematical discipline. And as it impinges on AI it joins in where hundreds of gigabytes strain current resources. Application areas that will have benefited from analytic derivation of the exact computation rules for computational chemistry and fluid dynamics and others are in a similar situation.

Jean Sammet observed early on that "expressions grow to exceed the space available" [83]. This has held true to a really astonishing extent! Even today, Gröbner Bases and Quantifier Elimination can have explosive memory demands even on problems of modest size.

It is becoming evident that knowing the history of computer algebra and symbolic computation is important for the general public, not just for us. By the time of this ISSAC meeting, the authors of this paper will have run a session on historical topics at the 30th Applications of Computer Algebra (ACA) conference in Crete in July 2025. The plan there is to have a number of first-hand reports from individuals who were present in earlier days, and then to form a group to build up a fuller record of this history in book form. Those who read this paper and who have relevant information or insight are encouraged to get in touch so that they can provide input to that enterprise.

[13] https://www.asc-csa.gc.ca/eng/canadarm/, http://www.maplesoft.com/view.aspx?SF=141144/ChangingFaceRobotic.pdf

[14] See https://en.wikipedia.org/wiki/Lisp_machine. See also the discussion at https://en.wikipedia.org/wiki/Symbolics and the adversarial connection to the modern GNU project.

[15] http://felix.unife.it/Root/d-Mathematics/d-The-mathematician/d-History-of-mathematics/t-History-of-computer-algebra

Acknowledgments

This project was begun in a conversation at a meeting of the Computational Epistemology Think Tank run by The Rotman Institute of Philosophy. We have relied significantly on the record kept by the ACM Digital Library. We are grateful to Bruno Buchberger for encouragement and comments on an earlier draft. We also thank the referees, and Michael Monagan, for their comments and suggestions, many of which were extremely helpful.

***Note Added in Proof**: Michael Monagan read the arXiv preprint of this paper and pointed out some additional references, including the work of Zassenhaus who in [101] converted Hensel's lemma into what is now known as "Hensel lifting," and other important work on multivariate GCD, such as [13], which was required to avoid the phenomenon of intermediate expression swell. Other suggestions of his will be taken into account for any future version of this paper.*

References

[1] Erwin H. Bareiss. 1968. Sylvester's identity and multistep integer-preserving Gaussian elimination. *Mathematics of computation* 22, 103 (1968), 565–578.
[2] Erwin H. Bareiss. 1972. Computational solutions of matrix problems over an integral domain. *IMA Journal of Applied Mathematics* 10, 1 (1972), 68–104.
[3] David Barton. 1967. A Scheme for Manipulative Algebra on a Computer. *Comput. J.* 9, 4 (02 1967), 340–344. https://doi.org/10.1093/comjnl/9.4.340
[4] David Barton, Stephen R. Bourne, and Colin J. Burgess. 1968. A simple algebra system. *Comput. J.* 11, 3 (01 1968), 293–298. https://doi.org/10.1093/comjnl/11.3.293
[5] David Barton and John P. Fitch. 1972. Applications of algebraic manipulation programs in physics. *Reports on Progress in Physics* 35, 1 (1972), 235.
[6] Michael Beeler, R. William Gosper, and Richard Schroeppel. 1972. *HAKMEM.* Technical Report AIM 239. MIT AI Lab.
[7] Bernice Bender. 1966. A computer system for algebra and analytic differentiation. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation (SYMSAC '66).* ACM, New York, NY, USA, 0201–0227. https://doi.org/10.1145/800005.807967
[8] Thomas J. (Tim) Bergin and Jean E. Sammet. 2006. Jean E. Sammet interview: March 28, April 4, April 11 and April 18, 2006. In *ACM Oral History Interviews.* ACM, New York, NY, USA, 78 pages. https://doi.org/10.1145/1141880.1243440
[9] Elwyn R. Berlekamp. 1967. Factoring Polynomials over Finite Fields. *Bell System Technical Journal* 46 (1967), 1853–1859.
[10] Myrna D Bernick, E. David Callender, and J. R. Sanford. 1961. ALGY—an algebraic manipulation program. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference.* ACM, New York, NY, USA, 389–392.
[11] Ann Boyle and Bobby F. Caviness. 1988. Future directions for research in symbolic computation. In *Report on Symbolic Algebraic Computations Workshop, April, TR-200.* Society for Industrial and Applied Mathematics, Washington DC, USA, 29–30.
[12] Claude Brezinski, Gérard Meurant, and Michela Redivo-Zaglia. 2022. *A Journey through the History of Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA. https://doi.org/10.1137/1.9781611977233
[13] William S Brown. 1971. On Euclid's algorithm and the computation of polynomial greatest common divisors. *Journal of the ACM (JACM)* 18, 4 (1971), 478–504.
[14] Bruno Buchberger. 1965. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* Ph. D. Dissertation. Mathematical Institute, Leopold Franzens University, Innsbruck, Austria.
[15] Bruno Buchberger. 1970. Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems. *Aequationes Mathematicae* 4, 3 (Oct. 1970), 374–383. https://doi.org/10.1007/bf01844169
[16] Bruno Buchberger. 1976. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.* 10, 3 (Aug. 1976), 19–29. https://doi.org/10.1145/1088216.1088219
[17] Bruno Buchberger. 1998. An Algorithmic Criterion for the Solvability of Algebraic Systems of Equations. In *Gröbner Bases and Applications*, Bruno Buchberger and Franz Winkler (Eds.). London Mathematical Society Lecture Note, Vol. 251. Cambridge University Press, Cambridge, UK, 535–545.
[18] 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 (2006), 475–511. https://doi.org/10.1016/j.jsc.2005.09.007 Logic, Mathematics and Computer Science: Interactions in honor of Bruno Buchberger (60th birthday).
[19] Bruno Buchberger, George E. Collins, Rüdiger Loos, and Rudolf Albrecht. 1982. Computer algebra symbolic and algebraic computation. *SIGSAM Bull.* 16, 4 (Nov. 1982), 5. https://doi.org/10.1145/1089310.1089312
[20] Bruno Buchberger, George E. Collins, and Rüdiger Loos with the cooperation of Rudolf Albrecht (Eds.). 1982. *Computer algebra — symbolic and algebraic computation.* Springer, Vienna. i–vi, 1–283 pages.
[21] Bobby F. Caviness. 1970. On canonical forms and simplification. *Journal of the ACM (JACM)* 17, 2 (1970), 385–396.
[22] Alonzo Church. 1936. Correction to A Note on the Entscheidungsproblem. *The Journal of Symbolic Logic* 1, 3 (1936), 101–102.
[23] Alonzo Church. 1936. A Note on the Entscheidungsproblem. *The Journal of Symbolic Logic* 1, 1 (1936), 40–41.
[24] George E. Collins. 1971. The SAC-1 system: An introduction and survey. In *SYMSAC'71: Proceedings of the second ACM symposium on Symbolic and algebraic manipulation.* ACM, New York, 144–152. https://doi.org/10.1145/800204.806279
[25] George E. Collins. 1974. Quantifier elimination for real closed fields by cylindrical algebraic decomposition--preliminary report. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)* 8, 3 (Aug. 1974), 80–90.
[26] George E Collins. 1976. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis. *ACM SIGSAM Bulletin* 10, 1 (1976), 10–12.
[27] ISSAC Steering Committee. 2025. Past ISSAC Conferences. https://www.issac-conference.org/past.php.
[28] Stephen A. Cook. 1966. *On the minimum computation time of functions.* Ph. D. Dissertation. Harvard University.
[29] Stephen A. Cook and Stál O. Aanderaa. 1969. On the minimum computation time of functions. *Trans. Amer. Math. Soc.* 142, 0 (1969), 291–314. https://doi.org/10.1090/s0002-9947-1969-0249212-8
[30] Elizabeth Cuthill. 1966. A FORMAC program for the solution of linear boundary and initial value problems. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation (SYMSAC '66).* ACM, New York, NY, USA, 0801–0850. https://doi.org/10.1145/800005.807958
[31] Elizabeth Cuthill and James McKee. 1969. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference (ACM '69).* ACM, New York, NY, USA, 157–172. https://doi.org/10.1145/800195.805928
[32] J. M. A. Danby, André Deprit, and A. R. M. Rom. 1966. The symbolic manipulation of Poisson series. In *Proceedings of the First ACM Symposium on Symbolic and Algebraic Manipulation (SYMSAC '66).* ACM, New York, NY, USA, 0901–0934. https://doi.org/10.1145/800005.807970
[33] J. H. Davenport, Y. Siret, and E. Tournier. 1993. *Computer Algebra, Systems and Algorithms for Algebraic Computation* (2nd ed.). Academic Press, USA.
[34] Amanda Davis. 2024. Jean Sammet, the accidental programmer. *IEEE Spectrum* 61 (December 2024). https://spectrum.ieee.org/jean-sammet-accidental-computer-programmer
[35] Martin Davis, Hilary Putnam, and Julia Robinson. 1961. The Decision Problem for Exponential Diophantine Equations. *The Annals of Mathematics* 74, 3 (Nov. 1961), 425–436. https://doi.org/10.2307/1970289
[36] Jean Della Dora, Claire Dicrescenzo, and Dominique Duval. 1985. *About a new method for computing in algebraic number fields.* Lecture Notes in Computer Science, Vol. 204. Springer, Berlin Heidelberg, 289–290. https://doi.org/10.1007/3-540-15984-3_279
[37] André Deprit. 1969. Canonical transformations depending on a small parameter. *Celestial mechanics* 1, 1 (1969), 12–30.
[38] Stephanie A. Dick. 2020. Coded conduct: making MACSYMA users and the automation of mathematics. *BJHS Themes* 5 (2020), 205–224. https://doi.org/10.1017/bjt.2020.10
[39] Claire Dicrescenzo and Dominique Duval. 1989. *Algebraic extensions and algebraic closure in Scratchpad II.* Lecture Notes in Computer Science, Vol. 358. Springer, Berlin Heidelberg, 440–446. https://doi.org/10.1007/3-540-51084-2_41
[40] John D. Dixon. 1982. Exact solution of linear equations using p-adic expansions. *Numer. Math.* 40, 1 (1982), 137–141.
[41] Carl Engelman. 1965. MATHLAB: a program for on-line machine assistance in symbolic computations. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part II: computers: their impact on society.* ACM, New York, 117–126.
[42] D.J. Farber, R.E. Grisold, and I.P. Polonsky. 1963. *A Preliminary Report on the String Manipulation Language SNOBOL.* Technical Report Unpublished Technical Memorandum 63-3344-2. Bell Telephone Laboratories.
[43] D.J. Farber, R.E. Grisold, and I.P. Polonsky. 1964. SNOBOL, a string manipulation language. *J. ACM* 11, 1 (1964), 21–30.
[44] Lawrence M Fisher. 2017. Jean E. Sammet 1928–2017. *Commun. ACM* 60, 7 (2017), 22–22.
[45] John P. Fitch. 2009. CAMAL 40 Years on - Is Small Still Beautiful?. In *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend, Canada, July 6-12, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5625)*, Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt (Eds.). Springer, Berlin, Heidelberg, 32–44. https://doi.org/10.1007/978-3-642-02614-0_8

[46] John P. Fitch, P. Herbert, and Arthur C. Norman. 1976. Design features of COBALG. In *Proceedings of the third ACM symposium on Symbolic and algebraic computation*. ACM, New York, 185–188.

[47] Xiao-Shan Gao. 2017. Wen-Tsun Wu: His Life and Legacy. *ACM Commun. Comput. Algebra* 51, 2 (Oct. 2017), 73–79. https://doi.org/10.1145/3151131.3151136

[48] W. Morven Gentleman. 1972. Optimal Multiplication Chains for Computing a Power of a Symbolic Polynomial. *Math. Comp.* 26, 120 (1972), 935–939.

[49] W. Morven Gentleman. 1974. Experience with truncated power series. *SIGSAM Bull.* 8, 3 (1974), 61–62. https://doi.org/10.1145/1086837.1086846

[50] E. Goto, T. Soma, N. Inada, T. Ida, M. Idesawa, K. Hiraki, M. Suzuki, K. Shimizu, and B. Philipov. 1982. Design of a Lisp machine - FLATS. In *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming* (Pittsburgh, Pennsylvania, USA) *(LFP '82)*. ACM, New York, NY, USA, 208–215. https://doi.org/10.1145/800068.802152

[51] Richard D. Greenblatt, Thomas F. Knight, John T. Holloway, and David A. Moon. 1980. A LISP machine. *SIGIR Forum* 15, 2 (March 1980), 137–138. https://doi.org/10.1145/1013881.802703

[52] James H. Griesmer and Richard D. Jenks. 1971. SCRATCHPAD/1—an interactive facility for symbolic mathematics. In *Proc. 2nd Symposium on Symbolic and Algebraic Manipulation*. ACM, New York, 45–53.

[53] Ralph E. Griswold. 1978. A History of the SNOBOL Programming Langauges. *ACM Sigplan Notices* 13, 8 (1978), 275–308.

[54] Walter Habicht. 1948. Zur inhomogenen Eliminationstheorie. *Commentarii Mathematici Helvetici* 21, 1 (Dec. 1948), 79–98. https://doi.org/10.1007/bf02568027

[55] James W. Hanson, Jane Shearin Caviness, and Camilla Joseph. 1962. Analytic differentiation by computer. *Commun. ACM* 5, 6 (June 1962), 349–355. https://doi.org/10.1145/367766.368195

[56] Anthony C. Hearn. 1966. Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer. *Comm. ACM* 9, 8 (1966), 573–577. https://doi.org/10.1145/365758.365766

[57] Anthony C. Hearn. 1968. REDUCE: A User-Oriented Interactive System for Algebraic Simplification. In *Interactive Systems for Experimental Applied Mathematics*, M. Klerer and J. Reinfelds (Eds.). Academic Press, New York, 79–90.

[58] Anthony C. Hearn. 2005. REDUCE: The First Forty Years. In *Algorithmic Algebra and Logic. Proceedings of the A3L*, Andreas Dolzmann, Andreas Seidl, and Thomas Sturm (Eds.). Books on Demand GmbH, Norderstedt, 19–24. http://reduce-algebra.com/reduce40.pdf

[59] Grete Hermann. 1926. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale: Unter Benutzung nachgelassener Sätze von K. Hentzelt. *Math. Ann.* 95, 1 (1926), 736–788.

[60] Grete Hermann. 1998. The question of finitely many steps in polynomial ideal theory. *SIGSAM Bull.* 32, 3 (Sept. 1998), 8–30. https://doi.org/10.1145/307339.307342

[61] A. Karatsuba and Yu. Ofman. 1962. Multiplication of many-digital numbers by automatic computers. *Dokl. Akad. Nauk SSSR* 145 (1962), 293–294. Issue 2.

[62] Donald E. Knuth. 1975. Son of seminumerical algorithms. *ACM SIGSAM Bulletin* 9, 4 (1975), 10–11.

[63] S. Y. Ku and R. J. Adler. 1969. Computing polynomial resultants: Bezout's determinant vs. Collins' reduced P.R.S. algorithm. *Commun. ACM* 12, 1 (Jan. 1969), 23–30. https://doi.org/10.1145/362835.362839

[64] C.Y. Lee et al. 1962. *A Language for Symbolic Communication*. Technical Report Unpublished Technical Memorandum 62-3344-4. Bell Telephone Laboratories.

[65] Steven Levy. 1984. *Hackers: Heroes of the Computer Revolution*. Anchor Press/Doubleday, USA. Google-Books-ID: o3YfAQAAIAAJ.

[66] John D. Lipson. 1971. Chinese remainder and interpolation algorithms. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*. ACM, New York, 372–391.

[67] Jan Łukasiewicz. 1931. *Uwagi o aksjomacie Nicod'a io „dedukcji uogólniającej"*. skł. gł. Księgarnia SA Książnica-Atlas, Lviv. https://sbc.org.pl/dlibra/publication/edition/18864

[68] John McCarthy. 1960. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. *Commun. ACM* 3, 4 (1960), 184–195. https://doi.org/10.1145/367177.367199

[69] S. Millman (Ed.). 1984. *A History of Engineering and Science in the Bell System: Communications sciences (1925-1980)*. AT & T Bell Laboratories, Murray Hill.

[70] Joel Moses. 2012. Macsyma: A personal history. , 123–130 pages.

[71] Joel Moses and David Y. Y. Yun. 1973. The EZ GCD algorithm. In *ACM'73: Proceedings of the ACM annual conference*. ACM, New York, 159–166. https://doi.org/10.1145/800192.805698

[72] Vanni Noferini and Alex Townsend. 2016. Numerical instability of resultant methods for multidimensional rootfinding. *SIAM J. Numer. Anal.* 54, 2 (2016), 719–743.

[73] Joachim von zur Gathen and Jürgen Gerhard. 2003. *Modern computer algebra*. Cambridge University Press, Cambridge, UK.

[74] Arthur C Norman. 1975. Computing with formal power series. *ACM Transactions on Mathematical Software (TOMS)* 1, 4 (1975), 346–356.

[75] Allen Reiter. 1965. *Automatic generation of Taylor coefficients (TAYLOR)*. Mathematics Research Center, United States Army, the University of Wisconsin, USA.

[76] Daniel Richardson. 1969. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic* 33, 4 (1969), 514–520.

[77] Robert H. Risch. 1969. The problem of integration in finite terms. *Trans. Amer. Math. Soc.* 139 (1969), 167–189.

[78] Robert H. Risch. 1970. The solution of the problem of integration in finite terms. *Bull. Amer. Math. Soc.* 76 (1970), 605–608.

[79] Maxwell Rosenlicht. 1968. Liouville's theorem on functions with elementary integrals. *Pacific J. Math.* 24, 1 (1968), 153–161.

[80] Maxwell Rosenlicht. 1972. Integration in finite terms. *The American Mathematical Monthly* 79, 9 (1972), 963–972.

[81] Jean E. Sammet. 1966. Survey of formula manipulation. *Commun. ACM* 9, 8 (Aug. 1966), 555–569. https://doi.org/10.1145/365758.365762

[82] Jean E. Sammet. 1967. Formula manipulation by computer. In *Advances in Computers*. Vol. 8. Academic Press (Elsevier), New York, 47–102.

[83] Jean E. Sammet. 1967. *Programming Languages: History and Fundamentals*. Prentice-Hall, Englewood Cliffs, New Jersey.

[84] Jean E. Sammet. 1972. Programming languages: history and future. *Commun. ACM* 15, 7 (July 1972), 601–610. https://doi.org/10.1145/361454.361485

[85] Jean E. Sammet. 1990s. personal communication.

[86] Tateaki Sasaki. 1979. An arbitrary precision real arithmetic package in REDUCE. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposium on Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings (Lecture Notes in Computer Science, Vol. 72)*, Edward W. Ng (Ed.). Springer, Berlin, Heidelberg, 358–368. https://doi.org/10.1007/3-540-09519-5_87

[87] Arnold Schönhage and Volker Strassen. 1971. Fast multiplication of large numbers. *Computing* 7 (1971), 281–292.

[88] James R. Slagle. 1961. *A heuristic program that solves symbolic integration problems in freshman calculus: symbolic automatic integrator (SAINT)*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[89] Cynthia Solomon, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L. Miller, Margaret Minsky, Artemis Papert, and Brian Silverman. 2020. History of LOGO. *Proceedings of the ACM on Programming Languages* 4, HOPL (2020), 1–66.

[90] Alma Steingart. 2023. *Axiomatics: Mathematical Thought and High Modernism*. University of Chicago Press, USA.

[91] Volker Strassen. 1969. Gaussian elimination is not optimal. *Numer. Math.* 13, 4 (1969), 354–356. https://doi.org/10.1007/BF02165411

[92] Alfred Tarski. 1948. A decision method for elementary algebra and geometry. In *Quantifier elimination and cylindrical algebraic decomposition*. Springer, Vienna, 24–84.

[93] Andrei L. Toom. 1963. The complexity of a scheme of functional elements simulating the multiplication of integers. *Doklady Akad. Nauk SSSR* 150, 3 (1963), 496–498.

[94] M. Pilar Vélez, Tomás Recio, and Carlos Ueno. 2022. Niagara Falls and the Origins of Computer Algebra. *Maple Transactions* 2, 1 (Sept. 2022), 14 pages. https://doi.org/10.5206/mt.v2i1.14362

[95] Martinus J. Veltman. 1963. *A CDC 6600 program for symbolic evaluation of algebraic expressions*. Technical Report. CERN. https://vsys.physics.lsa.umich.edu/schip-docs/CERN-Schoonschip-1967.pdf

[96] Paul S. Wang and Linda Preiss Rothschild. 1973. Factoring multivariate polynomials over the integers. *ACM SIGSAM Bulletin* , 28 (Dec. 1973), 21–29. https://doi.org/10.1145/1086814.1086819

[97] Paul S. Wang and Linda Preiss Rothschild. 1975. Factoring multivariate polynomials over the integers. *Math. Comp.* 29, 131 (1975), 935–950.

[98] Henry S. Warren. 2002. *Hacker's Delight*. Addison-Wesley Professional, Boston. Google-Books-ID: h3zvwQEACAAJ.

[99] Stephen M. Watt. 2009. On the Future of Computer Algebra Systems at the Threshold of 2010. In *Proceedings of the Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Computer Mathematics and Mathematical Aspects of Computer and Information Sciences (COE Lecture Notes, Vol. 22)*. Kyushu University, Japan, 422–430.

[100] David Y. Y. Yun. 1974. *The Hensel lemma in algebraic manipulation*. Ph. D. Dissertation. MIT.

[101] Hans Zassenhaus. 1969. On Hensel factorization I. *J. Number Theory* 1, 1 (1969), 291–311.