A history of efficiency problems in Maple

Michael Monagan
Department of Mathematics
Simon Fraser University
Burnaby, BC, Canada, V5A 1S6
mmonagan@sfu.ca

Abstract

The Maple project began in 1980 at the University of Waterloo. The most important design goal was that Maple be powerful, that is, Maple was efficient so that it could handle large inputs and Maple could solve a wide range of algebraic problems.

The early releases of Maple were not particularly efficient. This was not due to poor algebraic algorithms; rather, it had to do with the choice of the data representation for formulas, poor implementations of some systems algorithms, and design choices that resulted in a loss of efficiency. This talk presents seven efficiency problems that arose over Maple's history and some lessons we learned about writing efficient code in practice.

To assess whether today's Computer Algebra Systems are faster than those from the 1980s, we present a timing benchmark comparing the speed of Maxima with Maple, Magma and Singular on factoring determinants of matrices of polynomials.

1 Introduction

Development of Maple began in December 1980 at the University of Waterloo when a group of four professors, Bruce Char, Keith Geddes, Morven Gentleman and Gaston Gonnet decided to build their own Computer Algebra System. They chose to implement a kernel in the C language rather than Lisp, that supported a high level programming language suitable for implementing algebraic algorithms. The programming language was called Maple. It had to be efficient enough so that the algebraic algorithms (factoring polynomials, solving systems of equations, computing antiderivatives, etc.) could be implemented in it.

The first paper on Maple [1] was presented at the EUROCAL '83 conference in London, England in March 1983. The title of that first paper was "The Design of Maple: a compact, portable, and powerful Computer Algebra System." Keith Geddes presented the second paper on Maple [2] at the 1984 Macsyma Users conference in Schenectady, New York in July 1984. The title of the second paper was "On the Design and Performance of the Maple System".

I had joined the Maple group as a PhD student in the summer of 1983 having taken the Symbolic Computation course with Bruce Char in the winter of 1982 and completing my Masters degree in the Fall of 1982. The 1984 Macsyma Users conference was my first conference. The Maple group drove from Waterloo, Ontario to the conference in Schenectady, New York to show off what Maple

could do. At that time we regarded Macsyma as the system to beat. We showed that Maple was a lot faster than Macsyma and Reduce, at least on our benchmarks

The design of Maple was very successful in one aspect. Almost everyone (students and faculty) who was working on the Maple project at Waterloo in the 1980s was implementing algorithms in the Maple language. This resulted in a much faster development than would be possible if we had to implement algorithms in a systems language like C or Lisp. However, Maple really wasn't efficient. In this talk I share seven efficiency problems that arose over Maple's history and what we did to address them.

The first efficiency problem has to do with the cost of evaluating a formula by using assignment. The reason to present this problem is that of all the efficiency improvements that were made to Maple, this one change achieved the greatest overall speedup on Maple's test suite.

The second problem is a horrible efficiency bug in Maple and some lessons I learned about how not to find efficiency problems in Computer Algebra Systems.

The third problem is an efficiency problem that is present in all Computer Algebra Systems. We have $O(n^2)$ algorithms that should be $O(n)$ or $O(n \log n)$ in our systems. This leads to very slow running times for users. Users of Computer Algebra Systems are not computer scientists. They would not know how to test for a quadratic algorithm. They are unlikely to realize that the Computer Algebra System is using a quadratic algorithm and complain about it. So the quadratic algorithms remain in the systems for decades. I'll show an example.

The fourth problem is how to represent integers, in particular small integers, which appear often in formulas, for example as exponents in monomials like $x^2y^3z^2$. Early versions of Lisp [6] started encoding small integers in pointers to avoid allocating memory when doing arithmetic with small integers. Because Macsyma and Reduce were implemented in Lisp, they automatically benefited from this idea. Gaston Gonnet added this to Maple for Maple 6 in the 1990s. This resulted in another significant overall speedup on Maple's test suite.

The fifth efficiency problem was Maple's very slow numerical linear algebra library. Partly because of the dominance of Matlab in the engineering market, which is where the money is, Maplesoft redesigned Maple's linear algebra library from scratch for Maple 6. The new library includes compiled codes for hardware floating point precision. The new design also included a new user interface for vectors and matrices. David Hare of Maplesoft led the design effort.

The sixth problem is an inefficiency in polynomial division. The problem was solved by Johnson [3] in 1974 but since Johnson didn't benchmark his algorithm against those in the Computer Algebra Systems, the people who worked on Computer Algebra Systems missed the implication of his work; they were not embarrassed into fixing the problem!

The seventh problem concerns the representation used for polynomials. The choice matters because it affects the efficiency of multivariate polynomial arithmetic and that plays a significant role in the system's overall efficiency, for example in polynomial factorization. At the Macsyma Users conference in 1984, David Stoutemyer [7] showed that Pari/GP's recursive dense data representation was faster than the sparse distributed representation and the recursive sparse representation used by all other Computer Algebra Systems. This was a surprise because the conventional wisdom at the time was that one had to have a sparse representation for sparse polynomials.

Maple, Macsyma, and Mathematica use a very general representation for polynomials. They are slower at polynomial arithmetic than Magma and Singular which use a dedicated polynomial representation. Because Maple was so slow, Monagan and Pearce in [4, 5] designed a new data representation for polynomials in $\mathbb{Z}[x_1, x_2, \ldots, x_n]$ that was released in Maple 2013. Maple was now

the fastest system. In the talk I will present the polynomial data structure.

I end with a benchmark that tests the efficiency of factoring determinants of matrices of polynomials. Polynomial factorization is a success story in the history of Computer Algebra. We have good algorithms. But it's a huge amount of work to implement them. In a conversation with Tony Hearn, the principal author of Reduce, Tony told me that when they implemented polynomial factorization for Reduce, the "number of pages of code for Reduce doubled"!

How good are today's Computer Algebra Systems at polynomial factorization compared with older systems like Macsyma and Reduce? Fortunately we still have access to Maxima and Reduce so we can make such a comparison. I will compare polynomial factorization in Maxima with Maple, Magma and Singular. For benchmarks I compute and factor the $\det(V_n)$, $\det(T_n)$ and $\det(C_n)$ where V_n is the $n \times n$ Vandermonde matrix, T_n is the $n \times n$ symmetric Toeplitz matrix, and C_n is the $n \times n$ circulant matrix, all in the variables $x_1, x_2, \ldots, x_n$. For example

$$V_3 = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \quad T_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_1 & x_2 \\ x_3 & x_2 & x_1 \end{bmatrix} \quad C_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_3 & x_1 & x_2 \\ x_2 & x_3 & x_1 \end{bmatrix}$$

Their determinants factor as

$$\det(V_3) = (x_2 - x_1)(x_3 - x_1)(x_3 - x_2)$$

$$\det(T_3) = (x_1 - x_3)(x_1^2 + x_1 x_3 - 2x_2^2)$$

$$\det(C_3) = (x_1 + x_2 + x_3)(x_1^2 - x_1 x_2 - x_1 x_3 + x_2^2 - x_2 x_3 + x_3^2).$$

References

[1] Bruce W. Char, Keith O. Geddes, W. Morven Gentleman, Gaston H. Gonnet. The Design of Maple: a compact, portable, and powerful Computer Algebra System. Proceedings of EUROCAL '83, pp. 101–115, Springer, March 1983.

[2] Bruce Char, Gregory J. Fee, Keith O. Geddes, Gaston H. Gonnet, Michael B. Monagan, Stephen M. Watt. On the Design and Performance of the Maple System. Proceedings of the 1984 Macsyma User's Conference, pp. 189–220, July 1984.

[3] Johnson, S.C. Sparse polynomial arithmetic. ACM SIGSAM Bulletin **8**(3):63–71, 1974.

[4] Michael Monagan and Roman Pearce. POLY: A new polynomial data structure for Maple. In *Computer Mathematics*, Springer Verlag, pp. 325–348, October 2014.

[5] Michael Monagan and Roman Pearce. The design of Maple's sum-of-products and POLY data structures for representing mathematical objects. *Communications of Computer Algebra*, **48** (4), pp. 166–186, December 2014.

[6] Juho Snellman's Weblog. Numbers and tagged pointers in early Lisp implementations. https://www.snellman.net/blog/archive/2017-09-04-lisp-numbers/ Posted 2017.

[7] David Stoutemyer. Which polynomial representation is best? Surprises Abound! Proceedings of the 1984 Macsyma User's Conference, pp. 221–243, July 1984.
Accessible at https://udspace.udel.edu/items/55e9feab-ba45-46a1-9dc0-db2adbfb7157