

Efficiency Problems in the History of Maple

Michael Monagan
Simon Fraser University

History of Computer Algebra
ACA 2025

Early design goals of Maple

The Design of Maple: a compact, portable, and powerful Computer Algebra System.

[1] EUROCAL '83, London, England, March 1983.

Design a programming language that is good enough to implement most algebraic algorithms. The programming language was called Maple. Success!! 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. In July 1984 we drove to the Macsyma User's conference in Schenectady, New York, to show off Maple.

On the Design and Performance of the Maple System.

[2] Proceedings of the 1984 Macsyma User's Conference, pp. 189–220, 1984.

We were faster than Macsyma on a variety of tasks, at least on our benchmarks.
However, Maple really wasn't that efficient.
Six efficiency problems.

1: Evaluation.

Maple

```
> f := x+2*y+x*ln(x);  
f := x + 2 y + x ln(x)  
  
> x := 1;  
x := 1  
  
> f;  
1 + 2 y
```

Macsyma

```
(%i1) f : x+2*y+x*log(x);  
(%o1) 2 y + x log(x) + x  
(%i2) x : 1;  
(%o2) 1  
(%i3) f;  
(%o3) 2 y + x log(x) + x  
(%i4) ev(f);  
(%o4) 2 y + 1
```

The evaluation model that Maple uses is advertised as a very desirable feature.
Who got this right?

1: Evaluation.

```
> NumTermsInx := proc(f,x) local g,c,n,i;  
>   if type(f,'+') then g := [op(f)]; else g := [f]; fi;  
>   c := 0;  
>   n := nops(g);  
>   for i to n do if has(g[i],x) then c := c+1; fi; od;  
>   return c;  
> end;  
> f := x^3-3*x^2*y-3*y^2+5;
```

$$f := x^3 - 3x^2y - 3y^2 + 5$$

```
> NumTermsInx(f,y);
```

2

If f has n terms in m variables, the evaluation cost of $g[i]$ is $O(mn)$. Total is $O(mn^2)$!!

For Maple 4.0 I changed evaluation of local variables to be 1 level like parameters but globals still used full evaluation. No users complained. 10% gain!

2: A very bad hash function

$$V_3 = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \quad \det(V_3) = -x_1^2 x_2 + x_1^2 x_3 + x_1 x_2^2 - x_1 x_3^2 - x_2^2 x_3 + x_2 x_3^2.$$

In general $\det(V_n)$ has $n!$ terms.

I noticed that computing $\det(V_8)$ in Maple 3.3 was slow. Why?

2: Maple's unique representation model for algebraic expressions

Maple maintains a large hash table of all subexpressions so they are stored once. How?

> m1 := x*y^2*z^3; $m_1 =$

PROD	z	3	y	2	x	1
------	---	---	---	---	---	---

> m2 := z^3*y^2*x; $m_2 =$

PROD	x	1	y	2	z	3
------	---	---	---	---	---	---

The hash function must be commutative on x^1, y^2, z^3 so that $hash(m_2) = hash(m_1)$.

2: Maple's unique representation model for algebraic expressions

Maple maintains a large hash table of all subexpressions so they are stored once. How?

> m1 := x*y^2*z^3; $m_1 =$

PROD	z	3	y	2	x	1
------	---	---	---	---	---	---

> m2 := z^3*y^2*x; $m_2 =$

PROD	x	1	y	2	z	3
------	---	---	---	---	---	---

The hash function must be commutative on x^1, y^2, z^3 so that $hash(m_2) = hash(m_1)$.

> m3 := x^3*y^2*z; $m_3 =$

PROD	x	3	y	2	z	1
------	---	---	---	---	---	---

Since $m_1 \neq m_3$ we want $hash(m_3) \neq hash(m_1)$.

The blunder horribilis: hash was commutative on $x, 1, y, 2, z, 3$ so $hash(m_1) = hash(m_3)$!

It was easy for me to fix the problem. But how could this happen and why is it a disaster?

2: Maple's unique representation model for algebraic expressions

Consider V_4 the 4 by 4 Vandermonde matrix and its determinant D_4 .

Group 1: missing x_1	$-x_2^3 x_3^2 x_4, x_2^3 x_3 x_4^2, x_2^2 x_3^3 x_4, -x_2^2 x_3 x_4^3, -x_2 x_3^3 x_4^2, x_2 x_3^2 x_4^3$
Group 2: missing x_2	$x_1^3 x_3^2 x_4, -x_1^3 x_3 x_4^2, -x_1^2 x_3^3 x_4, x_1^2 x_3 x_4^3, x_1 x_3^3 x_4^2, -x_1 x_3^2 x_4^3$
Group 3: missing x_3	$-x_1^3 x_2^2 x_4, x_1^3 x_2 x_4^2, x_1^2 x_2^3 x_4, -x_1^2 x_2 x_4^3, -x_1 x_2^3 x_4^2, x_1 x_2^2 x_4^3$
Group 4: missing x_4	$x_1^3 x_2^2 x_3, -x_1^3 x_2 x_3^2, -x_1^2 x_2^3 x_3, x_1^2 x_2 x_3^3, x_1 x_2^3 x_3^2, -x_1 x_2^2 x_3^3$

Table: Terms in $\det(V_4)$, the 4×4 Vandermonde matrix

Each group has $n!/n = (n-1)!$ terms and each monomial in each group is in the same variables and the exponents are a permutation of 1, 2, 3. Thus every monomial in each group has the same hash value! This means searching Maple's `simp1` table for one of these monomials is $O((n-1)!)$ instead of $O(1)$.

3: Asymptotic blunders in systems codes

If an algorithm is $O(n)$ or $O(n \log n)$ and the programmers' implementation is $O(n^2)$ we say **the programmer has committed an asymptotic blunder**.

The bad hash function in Maple 3.3 is an example.

Do Maple, Magma, Singular, Macsyma have $O(n^2)$ algorithms in systems codes that should be $O(n)$?

3: Asymptotic blunders in systems codes

Read in a polynomial f with t terms in n variables from a text file.

```
f := 3*x^3-2*w*y*z-5*x*y^2*z+w^3-2*x*y*z-5*y*z^2*x+z^3-2*w*x*y;
```

It should be $O(nt \log t)$?

Maple 3.3 and Singular 3.1 were $O(nt^2)$.

Macsyma and Magma are still $O(nt^2)$!

	Maxima		Magma	Maple		Singular
t	5.45.0	(space)	V2.28-29	2024	(.m format)	3.4.1
2000	0.77		0.06	0.010		0.006
4000	2.93		0.12	0.019		0.012
8000	11.92	(10.8gb)	0.43	0.038	(0.003)	0.024
16000	47.72	(43.4gb)	1.47	0.075	(0.009)	0.049
32000	199.27	(171gb)	Seg fault	0.149	(0.012)	0.099
256000	NA	NA		1.799	(0.070)	0.924

CPU time (in seconds) to read in a polynomial f with t terms in 8 variables with 2 digit coefficients.

Why does it take years (decades!) for such problems to be identified and fixed?

4: Representation of small integers

Computer Algebra Systems allow integers to have arbitrary length. Before GMP, Maple stored long integers in arrays using a decimal base of $B = 10^4$ on a 32 bit computer.

INTPOS	3
--------	---

 and

INTNEG	3333	2222	11
--------	------	------	----

Maple 3.3's representation for the integers 3 and -1122223333 .

Most integer arithmetic in computations is with small integers.

PROD	$\uparrow x$	$\uparrow 1$	$\uparrow y$	$\uparrow 2$	$\uparrow z$	$\uparrow 3$
------	--------------	--------------	--------------	--------------	--------------	--------------

Maple 3.3's representation for the monomial xy^2z^3 .

Monomial multiplications are very slow.

How can we avoid allocating memory to add $2 + 3$?

4: Representation of small integers

PROD	$\uparrow x$	$\uparrow 1$	$\uparrow y$	$\uparrow 2$	$\uparrow z$	$\uparrow 3$
------	--------------	--------------	--------------	--------------	--------------	--------------

Maple 3.3's representation for the monomial xy^2z^3 .

Word pointers are even; the least significant 3 bits are 0.

For a 64 bit computer encode $-2^{62} \leq x < 2^{62}$ as $2x + 1$ which is odd.

PROD	$\uparrow x$	3	$\uparrow y$	5	$\uparrow z$	7
------	--------------	---	--------------	---	--------------	---

Maple 2024's representation for the monomial $x^1y^2z^3$.

In [7], Juho Snellman traces the idea back to early versions of Lisp.

But the Maple team did not know about it. We were C programmers!

Gaston Gonnet implemented it in Maple circa 1993. Another big efficiency gain.

5: Numerical Linear Algebra

Maple was also very slow at numerical linear algebra. Why?

- 1 The data representation used software floats.

FLOAT	↑ 314	↑ -2	FLOAT	629	-3
-------	-------	------	-------	-----	----

Maple's representation for $\pi = 3.14$ in Maple 3.3 and Maple 6.

- 2 Maple used a hash table to represent arrays, vectors and matrices.
- 3 The numerical library was coded in interpreted Maple not compiled C.

A culture change: Some algorithms needed to be implemented in C.

For Maple 6 arrays of hardware floats (singular, double and software precision) were added along with the LAPACK library. Why? To compete with Matlab.

David Hare of Maplesoft led the project.

5: Numerical Linear Algebra

$n = 200$	linalg	LinearAlgebra	speedup
solve $Ax = b$	8.35	0.0042	1988x
multiply AB	15.18	0.0142	1069x
singular values of A	18.06	0.0150	1204x
eigenvalues of A	6.81	0.0748	91x

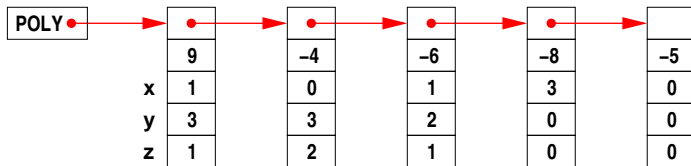
Timings in CPU seconds comparing the old linear algebra package `linalg` that uses software floats and is coded in Maple verses the

Why did this take so long?

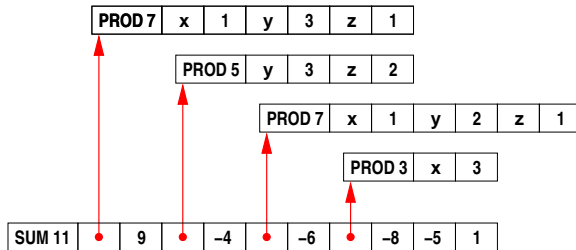
Maple was not embarassed by Macsyma into fixing it.

6: Polynomial Representations

Maple, Mathematica, Magma, Macsyma, and Singular use a distributed representation.



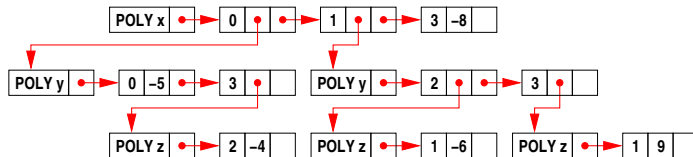
Singular's sum-of-products representation for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$



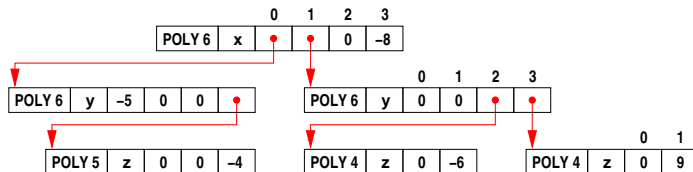
Maple's sum-of-terms representation for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$

6: Polynomial Representations

Reduce, Derive, Macsyma, Trip and Pari use recursive representations.



Trip's sparse recursive representation for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$



Pari's recursive dense representation for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$

6: Polynomial Representations

Which polynomial representation is best for \times and \div ?

6: Polynomial Representations

Which polynomial representation is best for \times and \div ?

At the Macsyma User's Conference in 1984, David Stoutemyer [8] observed that recursive dense is faster than recursive sparse which is faster than distributed.

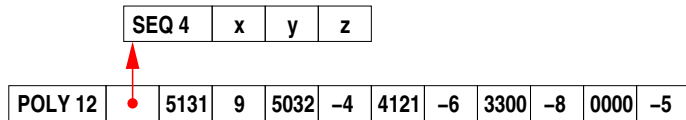
In [3] Richard Fateman in 2005 confirmed Stoutemyer's observations and also that Maple, Mathematica and Macsyma are slower than Singular and Magma.

Can the distributed representation be rescued ?

The problem is the monomial multiplications.

6: Polynomial Representations

Roman Pearce and Michael Monagan introduced POLY into Maple 2014 [4, 5].



Maple's POLY representation for $9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$
Encode $x^i y^j z^k$ as $k + 2^{16}j + 2^{32}i + 2^{48}(i + j + k)$, a 64 bit integer.

Encode exponents in $b = \lfloor 64/(n+1) \rfloor$ bits and $\deg(f)$ using $64 - nb$ bits if f is not linear, has integer coefficients, and all monomials fit, otherwise use old SUM of PROD.

- 1 Monomials use one word instead of $2n + 1$ words.
- 2 Monomial comparisons are 64 bit integer comparisons.
- 3 Monomial multiplication is a 64 bit integer addition.
- 4 Division does not need to test for overflow.

Three polynomial factorization benchmarks.

Has there been any real progress in polynomial factorization since 1980?

Compare Macsyma 5.45.0 with Maple 2024, Magma V2.28-19 and Singular 3.4.1

on three factorization benchmarks for $\mathbb{Z}[x_1, x_2, \dots, x_n]$, namely, factoring $\det(V_n)$, $\det(T_n)$ and $\det(C_n)$.

Macsyma and Magma use Paul Wang's multivariate Hensel lifting from 1978 [9].

Maple uses Monagan and Tuncer's random polynomial time algorithm from 2016 [6].

Singular uses Michael Lee's factorization code from his PhD Thesis in 2013.

Benchmark 1: factor Vandermonde determinants

$$V_3 = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \quad \det(V_3) = -x_1^2 x_2 + x_1^2 x_3 + x_1 x_2^2 - x_1 x_3^2 - x_2^2 x_3 + x_2 x_3^2$$
$$= (x_3 - x_2)(x_3 - x_1)(-x_1 + x_2).$$

		Maple		Magma		Singular		Maxima	
n	#det	det	factor	det	factor	det	factor	det	factor
7	5040	0.004	0.006	0.01	0.04	0.002	0.018	1.153	1.99
8	40320	0.024	0.036	0.02	0.50	0.029	0.180	17.56	44.99
9	362880	0.144	0.553	0.18	8.90	0.538	2.163	255.66	875.83
10	3628800	1.59	11.18	3.20	518.78	10.053	34.405	OM	NA
11	39916800	19.39	252.60	34.26	22,739.0	131.202	8,851.09	NA	NA
12	479001600	253.80	5334.6	NA	NA	NA	NA	NA	NA

Timings (in CPU seconds) to compute and factor $\det(V_n)$. OM = Out of Memory, NA = Not Attempted.

Benchmark 2: factor symmetric Toeplitz determinants

$$T_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_1 & x_2 \\ x_3 & x_2 & x_1 \end{bmatrix} \quad \det(T_3) = x_1^3 - 2x_1x_2^2 - x_1x_3^2 + 2x_2^2x_3 \\ = (x_1 - x_3)(x_1^2 + x_1x_3 - 2x_2^2).$$

		Magma		Maple		Singular		Maxima	
n	$\#f_1, \#f_2$	det	factor	det	factor	det	factor	det	factor
8	167,167	0.01	0.09	.008	.089	0.003	0.018	.840	40.44
9	294,153	0.08	0.26	.026	.218	0.019	0.150	7.93	896.7
10	931,931	0.64	1.50	.382	3.83	0.112	2.406	64.18	22,013.1
11	1730,849	5.09	4.55	1.52	9.71	0.695	29.249	373.2	NA
12	5579,5579	32.93	94.89	6.71	21.92	4.526	405.785	NA	NA
13	10611,4983	215.14	365.3	36.41	55.16	36.915	1,689.11	NA	NA
14	34937,34937	1204.37	5,484.3	169.2	388.80	130.86	96,242.9	NA	NA

Timings (in CPU seconds) to compute and factor $\det(T_n)$. NA = Not Attempted.









Benchmark 3: factor circulant matrix determinants

$$C_3 = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_2 & x_3 & x_1 \\ x_3 & x_1 & x_2 \end{bmatrix} \quad \det(C_3) = \begin{aligned} & -x_1^3 + 3x_1x_2x_3 - x_2^3 - x_3^3 \\ & = (x_1 + x_2 + x_3)(-x_1^2 + x_1x_2 + x_1x_3 - x_2^2 + x_2x_3 - x_3^2). \end{aligned}$$

		Magma		Maple		Singular		Maxima	
n	#det,#fmax	det	factor	det	factor	det	factor	det	factor
8	810, 86	0.01	0.05	.007	.084	0.003	0.018	0.580	0.361
9	2704, 1005	0.03	0.53	.027	.273	0.011	0.137	3.80	0.635
10	7492, 715	0.15	5.70	.135	2.18	0.056	0.340	30.39	4.037
11	32066,184756	0.95	104.52	.931	0.983	0.310	133.167	2922.1	35.42
12	86500, 621	7.02	2019.27	3.22	4.07	2.359	2.814	OM	113.97
13	400024,2704156	61.72	43,519.1	17.59	11.23	12.673	39,838.9	NA	NA
14	1366500,27132	427.74	> 6days	160.8	508.2	54.865	296.051	NA	NA

Timings (in CPU seconds) to compute and factor $\det(C_n)$. OM = Out of Memory, NA = Not Attempted.

References

-  Bruce Char, Keith Geddes, Morven Gentleman, Gaston Gonnet. The Design of Maple: a compact, portable, and powerful Computer Algebra System. Proceedings of EUROCAL '83, pp. 101–115, Springer, March 1983.
-  Bruce Char, Gregory Fee, Keith Geddes, Gaston Gonnet, Michael Monagan, Stephen Watt. On the Design and Performance of the Maple System. Proceedings of the 1984 Macsyma User's Conference, pp. 189–220, 1984.
-  Richard Fateman. Comparing the speed of programs for sparse polynomial multiplication. ACM SIGSAM Bulletin **37**(1):4–15, 2003.
-  Michael Monagan and Roman Pearce. POLY: A new polynomial data structure for Maple. In *Computer Mathematics*, Springer Verlag, pp. 325–348, October 2014.
-  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.
-  Michael Monagan and Baris Tuncer. Using Sparse Interpolation in Hensel Lifting. *Proceedings of CASC 2016*, LNCS **9890**:381–400, Springer, 2016.
-  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.
-  David Stoutemyer. Which polynomial representation is best? Surprises Abound! Proceedings of the 1984 Macsyma User's Conference, pp. 221–243, July 1984.
-  Paul S. Wang. An improved Multivariate Polynomial Factoring Algorithm. *Mathematics of Computation*, **32**, 1978.