POLY : A new polynomial data structure for Maple 17 *

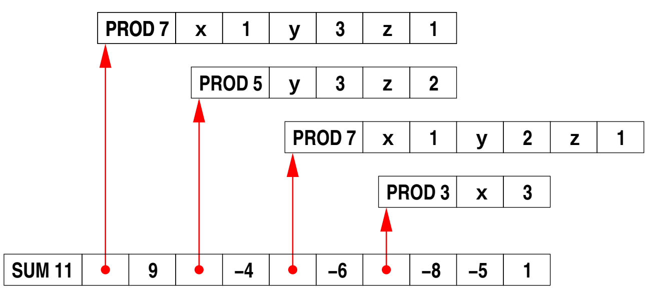
Michael Monagan and Roman Pearce
Department of Mathematics, Simon Fraser University
Burnaby B.C. V5A 1S6, Canada

Abstract

We demonstrate how a new data structure for sparse distributed polynomials in the Maple kernel significantly accelerates a large subset of Maple library routines. The POLY data structure and its associated kernel operations (degree, coeff, subs, has, diff, eval, ...) are programmed for high scalability, allowing polynomials to have hundreds of millions of terms, and very low overhead, increasing parallel speedup in existing routines and improving the performance of high level Maple library routines.

1 Introduction

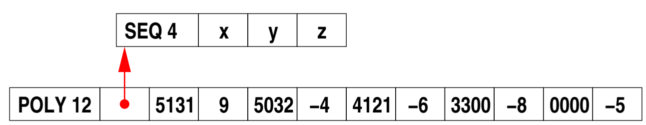
The figure on the left below shows the default polynomial data structure in Maple 16 and all previous versions. It is a "sum-of-products" where each term has a separate Maple object, a PROD, to represent the monomial. To compute the degree, a coefficient in x, test for a subexpression, or do almost anything else, the Maple kernel must descend through multiple levels of dags with recursive programs. This involves extensive branching and random memory access, both of which are slow.



The old sum-of-products representation
has irregular Maple dags for each term.

Representations for the polynomial

$$9\,xy^3z - 4\,y^3z^2 - 6\,xy^2z - 8\,x^3 - 5.$$



The new packed distributed representation
uses bit fields and sorts terms by total degree.

The figure on the right shows our new data structure for sparse distributed polynomials. The first word is a pointer to the variables which are sorted in Maple's canonical ordering for sets. This is followed by monomials and coefficients where the monomials encode the exponents together with the total degree in a single machine word. E.g. for xy^2z^3 we store the values $(6, 1, 2, 3)$ as $6 \cdot 2^{48} + 2^{32} + 2 \cdot 2^{16} + 3$ on a 64-bit machine. The terms are sorted into graded lex order by comparing the monomials as unsigned integers. This gives a canonical representation for the polynomial.

Three advantages of this representation are readily apparent. First, it is compact. Polynomials use two words per term instead of $2n + 3$ words, where n is the number of variables. For polynomials in 3 variables we save a factor of four. Second, by explicitly storing the variables and sorting the terms, we can execute a large number of common Maple idioms in constant time, e.g. $degree(f)$, $indets(f)$ (extract the set of variables in f), $has(f, x)$, and $type(f, polynom)$. Third, for large polynomials we avoid creating

*This work was supported by Maplesoft and the MITACS NCE of Canada.

a lot of small Maple objects (the PRODs) each of which must be simplified by Maple's internal simplifier and then stored in Maple's `simpl` table, an internal hash table of all Maple objects. They fill the `simpl` table and slow down Maple's garbage collector.

Polynomials in our development version of Maple are automatically stored in the POLY representation. If f is a polynomial in n variables with total degree d, then f is stored in the POLY representation on a 64 bit computer if f has integer coefficients, $d > 1$ and $d < 2^b$ where $b = \lfloor 64/(n+1) \rfloor$. Otherwise it is stored in the "sum-of-products" representation. All conversions between representations are automatic and invisible to the Maple user.

2 Algorithms

The new representation has allowed us to write many high performance algorithms for the Maple kernel. In the old data structure, most operations are $O(nt)$, where n is the number of variables and t is the number of terms. Maple must examine the entire "sum-of-products" structure because its contents are unknown. In the new data structure, we can often avoid doing expensive operations on all of the terms. We measured the speedup on a polynomial with one million terms in three variables, constructed as $f := expand(mul(randpoly(i, degree = 100, dense), i = [x, y, z])):$ The cost for evaluation is added to the other commands if you are using Maple interactively.

command	description	Maple 16	new dag	speedup	notes
$f;$	evaluation	0.162 s	0.000 s	$\rightarrow O(n)$	evaluate the variables
$coeff(f, x, 20)$	coefficient of x^{20}	2.140 s	0.005 s	420x	binary search for univariate f
$coeffs(f, x)$	extract all coefficients in x	0.979 s	0.190 s	5x	reorder exponents and radix sort
$degree(f, x)$	degree in x	0.073 s	0.002 s	24x	stop early using monomial degree
$degree(f)$	total degree	0.175 s	0.000 s	$\rightarrow O(1)$	first term in polynomial
$diff(f, x)$	differentiate wrt x	0.956 s	0.031 s	30x	terms remain sorted
$eval(f, x = 6)$	compute $f(6, y, z)$	3.760 s	0.245 s	15x	use Horner form (recursively)
$expand(2xf)$	multiply by a term	1.190 s	0.054 s	22x	terms remain sorted
$has(f, x^{101})$	search for subexpression	0.040 s	0.002 s	20x	$O(n)$ for names, $O(\log t)$ for terms
$indets(f)$	set of indeterminates	0.060 s	0.000 s	$\rightarrow O(1)$	first word in dag
$lcoeff(f, x)$	leading coefficient in x	0.058 s	0.005 s	11x	stop early using monomial degree
$subs(x = y, f)$	replace variable	1.160 s	0.071 s	16x	combine exponents, sort, merge
$taylor(f, x, 50)$	Taylor series to $O(x^{50})$	0.668 s	0.076 s	9x	get coefficients in one pass
$type(f, polynom)$	type check	0.029 s	0.000 s	$\rightarrow O(n)$	type check the variables

To achieve these gains, we employ a bit-level programming style (9) to avoid branches and loops. For example, to compute the degree of a monomial $x^3y^5z^7$ in $\{x, z\}$, we would mask the exponents for x and z and sum all of the fields using a parallel-prefix algorithm, which is $O(\log n)$. This is illustrated below, for a 32-bit monomial.

monomial $x^3y^5z^7$	$\boxed{00001111\ 00000011\ 00000101\ 00000111}$
mask for $\{x, z\}$	00000000 11111111 00000000 11111111
sum fields of	00000000 00000011 00000000 00000111

We chose a graded ordering as the default rather than pure lexicographical ordering for two reasons. Firstly, the graded ordering is the more natural ordering for output and secondy, unlike lexicographical order, in a graded ordering, the division algorithm cannot cause an overflow of the exponents from one bit field to another. In the graded ordering, many of the above operations can still be done without need to sort the result. For example, consider our polynomial $f = 9xy^3z - 4y^3z^2 - 6xy^2z - 8x^3 - 5$. If we differentiate f with respect to x we obtain $f' = 9y^3z + 0 - 6y^2z - 24x^2 + 0$. Notice that the non-zero terms in the derivative are sorted in the graded ordering.

3 Benchmarks

What impact on Maple's performance does the new POLY dag have for high level computations? And since the new POLY dag reduces the sequential overhead of computing with polynomials in Maple, how does this improve parallel speedup? Do we see any parallel speedup for high level operations? We consider two problems; computing determinants of matrices of polynomials and factoring polynomials.

3.1 A determinant benchmark.

Our first high level benchmark computes the determinant of the $n \times n$ symmetric Toeplitz matrix A for $6 \leq n \leq 11$. This is a matrix in n variables $x_1, \ldots, x_n$ with x_i appearing along the i^{th} diagonal and i^{th} subdiagonal. We implemented the Bareiss algorithm (1) in Maple and Magma to compute $\det(A)$. At the k^{th} elimination step, ignoring pivoting, the Bareiss algorithm computes

$$A_{i,j} := \frac{A_{k,k}A_{i,j} - A_{i,k}A_{k,j}}{A_{k-1,k-1}} \quad \text{for} \quad i = k+1, \ldots, n \quad \text{and} \quad j = k+1, \ldots, n$$

where the division is exact. At the end of the algorithm $A_{n,n} = \pm \det(A)$. Thus the Bariess algorithm does a sequence of $O(n^3)$ polynomial multiplications and divisions which grow in size, the largest of which occurs at the last step when $k = n - 1$, $i = n$ and $j = n$.

In Maple 16, large polynomial multiplications and divisions are done by our external library. This includes our software for parallel polynomial multiplication and parallel polynomial division from (7; 8). Polynomials are converted from the old sum-of-products representation into our new POLY dag, and back. In our new Maple where the POLY dag is the default; the same library is used but there are now no conversions.

In the table below column #det is the number of terms in the determinant, which has total degree n. Column #num is the number of terms in $A_{n-1,n-1}A_{n,n} - A_{n,n-1}A_{n-1,n}$ which has degree $2n-2$ and is much larger than $\det(A)$. We used a quad core Intel Core i5 CPU @ 2.66 GHz running 64-bit Mac OS X. Timings are real times in seconds, not cpu times. On 4 cores, we achieve a factor of 3 to 4 speedup over Maple 16, which is huge. These gains are entirely from reducing the overhead of Maple data structures; there is no change to the polynomial arithmetic over Maple 16. The reduction of overhead increases parallel speedup to 2.59x, from 1.6x in Maple 16.

			Maple 13	Maple 14		Maple 16		new POLY dag		Magma 2.17
n	#det	#num	1 core	1 core	4 cores	1 core	4 cores	1 core	4 cores	1 core
6	120	575	0.015	0.010	0.010	0.008	0.009	0.002	0.002	0.000 s
7	427	3277	0.105	0.030	0.030	0.030	0.030	0.006	0.006	0.020 s
8	1628	21016	1.123	0.180	0.180	0.181	0.169	0.050	0.040	0.200 s
9	6090	128530	19.176	1.330	1.330	1.450	1.290	0.505	0.329	2.870 s
10	23797	813638	445.611	18.100	13.800	14.830	12.240	6.000	3.420	77.020 s
11	90296	5060172	–	217.020	145.800	151.200	94.340	88.430	34.140	2098.790 s

3.2 A factorization benchmark.

In our second benchmark we see a large gain in performance on polynomial factorization. To provide some perspective, we include timings for Magma, Singular (4), Mathematica, and Trip (2), a computer algebra system for celestial mechanics. We used an Intel Core i5 750 @ 2.66GHz and a Core i7 920 @ 2.66GHz which had identical times in Maple 16. These are 64-bit quad core cpus.

All of the times in the table below are real times, not cpu times, in seconds. We report two times for Trip. The (RS) time is for Trip's optimized recursive sparse polynomial data structure POLYV. The (RD) time is the optimized recursive dense data structure POLPV. Both use multiprecision rational coefficients and Trip's parallel routines (3). Both timings reported for Trip are for 4 cores.

	Maple 13	Maple 16		new POLY dag		Magma	Singular	Mathem	Trip 1.2	
		1 core	4 cores	1 core	4 cores	2.16-8	3.1	atica 7.0	(RS)	(RD)
multiply										
$p_1 := f_1(f_1+1)$	1.60	0.053	0.029	0.047	0.017	0.30	0.58	4.79	0.010	0.008
$p_2 := f_2(f_2+1)$	1.55	0.054	0.028	0.047	0.016	0.30	0.57	5.06	0.018	0.016
$p_3 := f_3(f_3+1)$	26.76	0.422	0.167	0.443	0.132	4.09	6.96	50.36	0.088	0.073
$p_4 := f_4(f_4+1)$	95.97	1.810	0.632	1.870	0.506	13.25	30.64	273.01	0.433	0.336
divide										
$q_1 := p_1/f_1$	1.53	0.053	0.026	0.048	0.017	0.36	0.42	6.09	0.200	0.122
$q_2 := p_2/f_2$	1.53	0.053	0.026	0.048	0.017	0.36	0.43	6.53	0.170	0.144
$q_3 := p_3/f_3$	24.74	0.440	0.162	0.449	0.138	4.31	3.98	46.39	1.676	0.950
$q_4 := p_4/f_4$	93.42	1.880	0.662	1.920	0.568	20.23	15.91	242.87	7.292	4.277
factor										
p_1 12341 terms	31.10	2.58	2.46	1.20	0.94	6.15	12.28	11.82		
p_2 12341 terms	296.32	2.86	2.74	1.36	1.09	6.81	23.67	64.31		
p_3 38711 terms	391.44	15.19	13.00	9.57	6.16	117.53	97.10	164.50		
p_4 135751 terms	2953.54	53.52	44.84	31.83	16.48	332.86	404.86	655.49		

$f_1 = (1+x+y+z)^{20}+1$ 1771 terms $f_2 = (1+x^2+y^2+z^2)^{20}+1$ 1771 terms $f_3 = (1+x+y+z)^{30}+1$ 5456 terms $f_4 = (1+x+y+z+t)^{20}+1$ 10626 terms

The Maple timings are for executing the commands *p1 := expand(f1*(f1+1))*, *divide(p1,f1,'q1')* and *factor(p1)*. The improvement from Maple 13 to Maple 16 is due to our improvements to polynomial multiplication and division in (6; 7; 8) which we reported at ISSAC 2010 in (5). This is because most of the time in multivariate factorization was spent in "Hensel lifting" which consists of many polynomial multiplications and some exact divisions. However, there is little parallel speedup. We achieve significant additional speedup (compare Maple 16 with the new POLY dag) with the POLY dag used by default. Sequential speedup for factoring p_1 is a factor of $2.58/1.20 = 2.15$x and parallel speedup for factoring p_4 improved from a factor of $53.52/44.48 = 1.19$x in Maple 16 to $31.83/16.48 = 1.93$x in our new Maple.

References

[1] E. Bariess. Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination. *Mathematics of computation* **22** (102): 565−578, 1968.

[2] Gastineau, M., Laskar, J. Development of TRIP: Fast Sparse Multivariate Polynomial Multiplication Using Burst Tries. *Proceedings of ICCS 2006*, Springer LNCS **3992**, pp. 446–453, 2006.

[3] Gastineau, M. Parallel operations of sparse polynomials on multicores - I. Multiplication and Poisson bracket. *Proceedings of PASCO '2010*, ACM Press, pp. 44–52, 2010.

[4] Greuel, G.-M., Pfister, G., Schönemann, H., 2005. Singular 3.0: A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern, http://www.singular.uni-kl.de.

[5] M. Monagan, R. Pearce. Sparse Polynomial Multiplication and Division in Maple 14. *Communications in Computer Algebra*, **44**:4, 205–209, December 2010.

[6] Monagan, M., Pearce, R. Sparse Polynomial Division using Heaps. *J. Symb. Cmpt.* **46** (7) 807–822, 2011.

[7] Monagan, M., Pearce, R. Parallel Sparse Polynomial Multiplication Using Heaps. *Proceedings of ISSAC 2009*, ACM Press, pp. 295–315, 2009.

[8] Monagan, M., Pearce, R. Parallel Sparse Polynomial Division Using Heaps. *Proceedings of PASCO 2010*, ACM Press, pp. 105–111, 2010.

[9] Warren, Henry S. Hacker's Delight. Addison-Wesley, 2003.