Fermat Benchmarks For Rational Expressions in Maple

Michael Monagan Roman Pearce

Introduction

Maple's `normal` command simplifies rational expressions by placing the expression over a common denominator and cancelling a gcd. The first step is done by the kernel, which recognizes common factors by their address in memory.

$$\frac{1}{x(y+1)} + \frac{1}{y(y+1)} \to \frac{x+y}{xy(y+1)}$$

Maple deals with factored forms efficiently, but it does not compute gcds so the common denominator it uses may not be the lcm. For the large multivariate examples considered here this leads to blowup.

However the approach works well in most cases, including small problems, univariate problems, and expressions with functions. Maple's gcd code is interpreted, so the overhead of calling it many times could easily outweigh the savings.

In the end, we decided to augment the existing code with a preprocessing step for large problems. This approach has a lot of flexibility if something turns out to be inefficient.

The Fermat Benchmarks

The Fermat tests are a set of problems posted online by the author of Fermat [1]. The first one substitutes multivariate rational expressions into a multivariate polynomial and the result simplifies to zero. A smaller version of the problem removes some variables by evaluation first.

Maple is slow on both problems because the denominators have a non-trivial multivariate gcd. The large version has a gcd with 12 variables and 840 terms. Because Maple does not compute lcms, it multiplies by extra *powers* of the gcd before combining like terms. These polynomials are huge.

This example shows how to get an extra factor of g^2 in the numerator (assuming bg and dg are expanded).

$$\left(\frac{a}{b\,g}\right)^2 + \left(\frac{c}{d\,g}\right)^2 \to \frac{a^2\,(d\,g)^2 + c^2\,(b\,g)^2}{(b\,g)^2\,(d\,g)^2}$$

Factoring Subexpressions

Our first idea was to convert expanded polynomials to factored form since Maple is efficient in that case. The `subsindets` command transforms expressions of a given type (expanded sums) using a procedure (factor).

```
> f := subsindets(f, And('+',expanded), factor):
> normal(f);
```

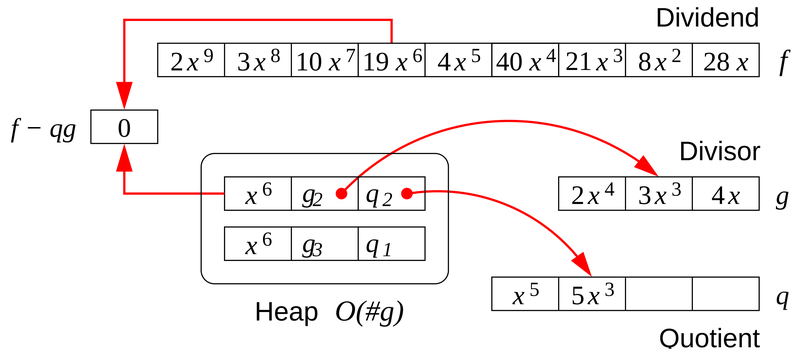
This change alone produced competitive timings (see benchmark table) although we observe high memory usage on the large Fermat test. We tried replacing each factor by a new variable. This simplifies quickly, and the challenge is to evaluate the numerator at a set of multivariate polynomials and expand the result.

Maple expands most objects recursively, which causes two problems in our case. The terms in a sum like f^2g^2 or f^2g^3 are expanded separately and merged, which uses a lot of temporary space, and powers are expanded first, so f^2g^2 is computed as $(f^2)\cdot(g^2)$. For sparse polynomials it is much faster to multiply $((f\cdot f)\cdot g)\cdot g$.

Substitution by Division

To address these inefficiencies we wrote a C routine to divide sparse multivariate polynomials by a set of sparse multivariate polynomials. If $\{\alpha_1, \alpha_2, \dots\}$ are new variables representing the factors $\{f_1, f_2, \dots\}$, then we can divide the numerator by $\{\alpha_1 - f_1, \alpha - f_2, \dots\}$ to substitute for the α_i's and expand. This algorithm is equivalent to expanding by repeated multiplication.

We can implement the division efficiently by using a heap of pointers as follows. For each term in a divisor, we store the product of that term with the next term of the quotient. Products are compared in the heap and merged in descending order. After a product is merged, the pointer for that divisor term increments to the next term of the quotient. The setup is shown below for one divisor.



Benchmarks

We compared our approach (factor + rem) running on Maple 2016 to Maple's normal command, Magma 2.21-12, and Fermat 5.21 on a Core i7 3930k 3.2 GHz with 64 GB of RAM with 64-bit Linux. Factoring produces acceptable performance, but our division with remainder is worth another order magnitude on the large problem. Our division routine can expand the numerator even if factoring is not performed, even though it suffers from blowup.

	Fermat Test #1 Large		Fermat Test #1 Small	
	time	memory	time	memory
normal	>10 hours	>64.0 GB	131.400 s	8.8 GB
rem only	80.550 s	2.5 GB	2.900 s	145.0 MB
factor only	4.700 s	0.9 GB	0.152 s	37.9 MB
factor + rem	0.213 s	11.5 MB	0.058 s	4.3 MB
Magma	9.130 s	223.0 MB	0.750 s	64.1 MB
Fermat	3.980 s	260.0 MB	0.380 s	30.6 MB

Conclusion

This code should be integrated into Maple's normal command in a future release. We also developed a heuristic to anticipate blowup so that Maple can call our code. Factoring univariate polynomials was found to be inefficient at present. We plan to use our division routine to improve Maple's expand command in the future.

References

[1] Robert H. Lewis. Take The Fermat Tests. http://home.bway.net/lewis/fermat/FerTest.html

[2] M. Monagan, R. Pearce. Sparse Polynomial Division Using a Heap. Journal of Symbolic Computation, 46 (7), 807–922, 2011.

NSERC CRSNG

Maplesoft Mathematics • Modeling • Simulation

cecm