(Extended Abstract)

Recent Developments in Core Library

Chee Yap and Chen Li
Department of Computer Science
Courant Institute, New York University
email: {yap,chenli}@cs.nyu.edu

October 6, 2000

1 Introduction

There is considerable interest in robust implementation of geometric algorithms. The Exact Geometric Computation (EGC) approach to robustness has the advantage that its basic techniques can be encoded in a general-purpose numerical library that is transparent to most users. This approach has been taken by the LEDA Library and our Core Library [2]. A fundamental task in such libraries is to determine the exact sign of a constant algebraic expression E. Typically, E is constructed from rational numbers, the four rational operations $(+, -, \times, \div)$ and square-root. Such expressions are called **radical expressions**. The sign determination task reduces to first finding some root bound b for E. We call $b > 0$ a **root bound** for E if the following holds: if $E \neq 0$ then $|E| \geq b$. The corresponding **root bit-bound** is defined to be $-\lg(b)$. While many classical root bounds are known, they are not always "constructive" in the sense of being easily computed from the structure of E. Until recently, the best constructive bound for division-free radical expressions is from Burnikel et al (BFMS) [1]. In fact, the BFMS bound is essentially optimal. However, for general radical expressions with division, their root bit-bound is quadratic in the algebraic degree of E. This can be a serious penalty in practice, since expressions with division are common (for instance, floating point input values in expressions will be represented by a division of two integers). In [3], we show a new constructive root bound that has linear dependence on the algebraic degree.

The new root bound has been implemented in the latest version of the Core Library (version 1.3), and significant speedups were achieved in various classes of expressions. We also mention several other improvements in this library.

2 Timings

The new constructive root bound has been implemented in our Core Library. Our experiments, based on version 1.2x of the Core library, compare the performances of our new bound and the BFMS bound. The timings, in seconds, are performed on a Sun Ultra 250 with two UltraSPARC-II 296 MHz CPUs and 512MB main memory.

1. The critical test in Fortune's sweepline algorithm is to determine the sign of the expression

$$E = \frac{a + \sqrt{b}}{d} - \frac{a' + \sqrt{b'}}{d'} \tag{1}$$

where a (or a'), b (or b') and d (or d') are $3L$-, $6L$- and $2L$-bit integers, respectively. Our root bit-bound improves the BFMS bound from $(79L + 30)$ bits to to $(19L + 9)$ bits. We generate some random inputs with different L values which always make $E = 0$, and put the timings of the tests in Table 1. We also converted the Fortune's implementation of this algorithm to use Core library. We ran the program on two kinds of inputs: (1) First we test on a non-degenerate data set (100 random points provided n Fortune's code distribution). The time for our new bound is 7.87 seconds while the BFMS bound takes 7.96 seconds.

Table 1: Timings for Fortune's expression in (1)

L	10	20	30	50	100	200
BFMS	0.08	0.52	2.51	4.06	28.1	218.76
NEW	0.02	0.08	0.22	0.32	1.72	11.73

This is not unexpected, since as explained in the introduction, our Core Library exploits the progressive evaluation technique, and the signs of Fortune's predicate on non-degenerate inputs can be determined without reaching the root bound. Thus the root bound has no effect on the complexity. (2) We use degenerate inputs comprising points on a (32×32) uniform grid with L-bit coordinates. The timings are reported in Table 2.

Table 2: Timings for Fortune's algorithm on degenerate inputs

L	10	20	30	50
BFMS	212.02	1929.27	2830.90	9948.63
NEW	83.90	217.37	232.99	404.11

2. The second test is to verify an expression which is identically zero. Let $x = \frac{a}{b}$ and $y = \frac{c}{d}$ (a, b, c, d are L-bit integers),

$$E = (\sqrt{x} + \sqrt{y}) - \sqrt{x + y + 2\sqrt{xy}}.$$

Note that divisions are introduced in the definitions of x and y. Our new bound requires computing $(40L + 38)$ bits, while the BFMS requires $(640L + 510)$ bits. The timings are in Table 3.

Table 3: Timings for Example 2

L	5	10	20	30
BFMS	1830.61	8768.29	$>$ 3hrs	$>$ 3hrs
NEW	0.71	3.17	17.04	55.43

In comparing the timings of BFMS with our new method, it is the *relative* speedup that should be stressed. Similar relative improvements should show up if the tests were made in other systems (e.g., `LEDA`).

3 Other Improvements

Another source of speedup in the latest Core Library is our incorporation of `LiDIA/CLN`'s bigNumber kernel. `CLN` has one of the fastest big Number packages (even when compared to the latest `gmp`, Version 3.1). As a rough gauge of the improvements brought about by the new root bound and the better kernel, we timed all the sample programs in the distribution. On a Sun UltraSPARC 10, 440 MHz machine, the time is 55.0 seconds under CORE 1.3, compared to 1040.38 seconds under CORE 1.2.2.

Numerical precision is a critical issue for our library, and hence numerical input and output need to be unambiguous and conveniently controllable by the user. The latest library incorporates several such improvements. In particular, because we keep track of error bounds, we can ensure that only significant digits are printed. User has a choice of scientific or positional formats for input and output.

References

[1] C. Burnikel, R. Fleischer, K. Mehlhorn, and S. Schirra. A strong and easily computable separation bound for arithmetic expressi ons involving radicals. *Algorithmica*, 27:87–99, 2000.

[2] V. Karamcheti, C. Li, I. Pechtchanski, and C. Yap. A Core library for robust numerical and geometric libraries. In *15th ACM Symp. Computational Geometry*, pages 351–359, 1999. Download, `ftp://cs.nyu.edu/pub/local/yap/exact/core.ps.gz`.

[3] C. Li and C. Yap. A new constructive root bound for algebraic expressions. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Jan. 2001. To appear. Preprint: `ftp://cs.nyu.edu/pub/local/yap/exact/`.