(extended abstract)

A New Number Core for Robust Numerical and Geometric Libraries*

Chee K. Yap
Department of Computer Science
Courant Institute
New York University

October 16, 1998

Abstract

We describe a new numerical core that can serve as the basis for robust numerical and geometric libraries. A novel feature of core is its hierarchy of numerical accuracies which can be accessed simultaneously by a conventional `C/C++` program.

We propose to build a **core library** (CORE) of critical geometric and numerical primitives around this numerical core. This library will be portable, efficient, robust and easy-to-use. The portability and efficiency are based on a state-of-the-art compiler technology (TRIMARAN) that can produce optimized code for a wide range of hardware architectures, particularly the new EPIC (explicitly parallel instruction computing) technology heralded by the Intel Merced chip. Numerical robustness is based on a scientifically sound approach called **exact geometric computation**. Ease-of-use is based on the possibility of using CORE with minimal change in programmer's behavior. Our library will, for the first time, make very powerful robustness techniques widely accessible to general users.

This talk describes current work on this library (with Professor Krishna Palem) and outline the practical as well as theoretical issues.

1 NONROBUSTNESS and GEOMETRIC EXACTNESS

Numerical nonrobustness is a widely known problem in all areas of numerical computing: computational sciences, simulation, engineering, modeling and manufacturing. In simple terms, nonrobustness is the property of a system which causes it crash for "mysterious reasons". We distinguish two kinds of numerical errors. **Quantitative errors** are usually the benign kind that are inevitable when we approximate numbers. But nonrobustness is symptomatic of a deeper phenomenon which we call **qualitative errors**. Such errors causes fundamental inconsistencies in the programming logic, leading to catastrophic errors.

Numerical analysts have much to teach us about good numerical computing practices. Their response to the nonrobustness is to suggest (1) the use of numerically stable algorithms and (2) to avoid of ill-conditioned inputs. "Stable" algorithms[1] is clearly useful in practice. But such algorithms only reduce the frequency of catastrophic errors but cannot eliminate it completely. Similarly, numerical analysts are right in pointing that catastrophic errors occur only for ill-conditioned problems. Unfortunately, in geometric computation, some inputs are deliberately ill-conditioned: collinearity of three points and parallel lines are perfectly normal occurrences in engineering designs.

Most attempts to treat nonrobustness begin with the assumption that a solution can be found within the fixed-precision computation (which is the main computational paradigm in current scientific computation). For a variety of reasons, these solutions have not been satisfactory. The basis of our approach is the **Exact Geometric Computation** (EGC). See [10] for a survey. EGC can, in principle, eliminate qualitative errors in the class of "algebraic" problems. The main challenge is to make EGC solutions efficient, or, "efficient enough", so that a user would prefer robust EGC solutions over a fast but nonrobust one. Recent papers

*Abstract of Invited Talk at 3rd CGC Workshop on Computational Geometry, Brown University, October 11-12, 1998.

[1]This concept is often informal, but see recent book of Trefethen and Bau [9] for a definition

have strongly suggested that EGC is practical for many basic problems in computational geometry. Nonlinear geometry remains a significant challenge. Current research is pushing the envelope of what can be made practical within the EGC approach. A number of effective tools and techniques (floating point filters, low degree predicates, precision-driven techniques, etc) are being developed for this purpose.

What is critically needed is an infrastructure in which such techniques can be (a) brought together together and (b) made easily accessible to all programmers, not just to experts in this research area. Today, a floating point package is a *sine qua non* for numerical applications. Similarly, we believe that a new numerical "core" is needed to support the EGC approach. This talk describes such a proposal.

We should note related efforts to construct major libraries of efficient and robust (EGC) geometric algorithms and data structures in the European community: `LEDA` [1] and `CGAL` [2]. Our goals are somewhat orthogonal to these, as we emphasize our library service in terms of facilities in support of EGC computing.

2 CORE LEVELS OF ACCURACY

A key feature of our CORE its ability to deliver very powerful techniques in robust computation, literally at a flip of a switch. From a "user view" this capability is encapsulated in four **levels of accuracy**:

I. **Machine Accuracy.** This is the conventional IEEE-standard [7].

II. **Arbitrary Accuracy.** No overflow or underflow occurs until some specified accuracy (say 128 bits) is exceeded.

III. **Guaranteed Accuracy.** The computed value of variable is guaranteed to some user-specified accuracy. The default accuracy is one bit of **relative precision**[2], which guarantees the correct sign of computed quantities.

IV. **Mixed Accuracy.** Each numerical quantity will have one of the previous three levels of accuracy. Level IV allows these 3 levels of accuracy to occur simultaneously in a computation, and serves as a tool for efficient control of accuracy.

A major design goal is to allow users to access these CORE capabilities with almost no change in programming behavior. These levels are "simultaneously" available to any program in the supported conventional languages (currently, `C/C++`). Users only have to add a simple preamble to their program. The simplest preamble is:

```
#define AccuracyLevel N        /* N=1,2,3 or 4 */
#include "CORE.h"
```

The only system which currently support level III accuracy for a class of non-rational expressions is our `Real/Expr` package [4, 6]. The supported expressions involve $+, -, \times, \div, \sqrt{\cdot}$. As a proof-of-concept, we[3] have already constructed a wrapper around the `Real/Expr` package so that level I, II and III accuracies can already be accessed by any `C/C++` program.

Although level III guarantees geometric exactness, for many applications, level II suffices. Even for the speed-conscious user, Level III can play a very useful role: it can validate the logic of their code. Level III accuracy is the key innovation of CORE. Its distinction from level II may not be obvious. In any computer algebra system (e.g, `Maple`), one can get level II accuracy. However, specifying "500 bits of accuracy" does not mean that all the 500 bits in a quantity are significant. On the other hand, guaranteeing 10 bits at level III may entail computing intermediate quantities to arbitrarily high precision.

3 EFFICIENCY ISSUES

We address efficiency and portability issues of this library development at three levels: (1) algorithm design level, (2) their realization as high-level language code, and (3) the eventual object code. While levels (1) and

[2] The user could also specify absolute precision bounds, or a combination of both.

[3] Thanks to our student Igor Pechtchanski. Among other things, the wrapper needed to remove all traces of non-conventional semantics in `Real/Expr`.

(2) can give asymptotic speedups, level (3) optimization can be critical in many applications. To appreciate our level (3) optimization effort, one needs to understand current developments in microprocessor technology.

The power of optimizing compilers has advanced by leaps and bounds in the last decade, spurred on by advances in **instruction level parallel** (ILP) microprocessors[4] that can execute multiple instructions in one cycle. The recently announced Intel IA64 (code name MERCED) takes this idea to a new plateau: Intel Corporation uses the term **explicitly parallel instruction computers** (EPIC) to describe such a CPU mode wherein the compiler controls the hardware resources and their allocation to instructions. To uncover the inherent instruction level parallelism in a `C` or `C++` program *and* to exploit it on an EPIC processor requires the full-blown and aggressive application of state-of-the-art optimizing compilers.

Our library work will exploit the TRIMARAN system, a comprehensive infrastructure for research in EPIC style processors and their optimizing compilers. Based on over 100 person-years of innovation, it is a collaborative effort between Hewlett-Packard laboratories, The University of Illinois's IMPACT project and NYU's REACT-ILP project [5]. The consortium officially released the TRIMARAN system into the public domain this summer. The main assets of the TRIMARAN system include a mechanism for describing microprocessors via a language called `HMDES` [3], a detailed simulation environment that is automatically produced for any `HMDES`-described machine, and a complete suite of optimizations that are also driven by the machine description. Users can add optimization modules to TRIMARAN – we can add EGC or CORE-specific optimizers, for instance.

4 FINAL REMARKS

We briefly mention some issues and current work.

We know how to deliver level III accuracy for algebraic computations. Extensions to elementary functions is completely open. For instance, just adding the function $\sin(x)$ and the constant π leads to problems that are closely related to undecidable problems [8]. In view of this, we define a level II.5, in which the system produces **certified accuracies** whenever possible.

Our library automatically deploys various EGC techniques (floating point filters, adaptive precisions computation, etc). We are exploring implementations that combine these with compiler-based techniques such as multi-threading and cache management.

We plan to define several **core library extensions** (COREX's) in which domain specific knowledge are used. Actual applications will be built on top of such COREX's. In this context, EGC opens up the new class of **geometric rounding problems** that are critical in applications: given a (consistent) geometric structure, to round this to some lower precision (still consistent) geometric structure.

To conclusion, our core library represents a new delivery system for bringing powerful techniques to bear on the non-robustness problem. In contrast to other work, we have chosen to focus on the numerical core which supports EGC. Another distinguishing mark is our attempt to achieve efficiency, not only at the algorithmic level, but also at the compiler front- and back-ends.

References

[1] C. Burnikel, J. Könnemann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig. Exact geometric computation in LEDA. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages C18–C19, 1995.

[2] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schoenherr. The CGAL kernel: a basis for geometric computation. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, pages 191–202, Berlin, 1996. Springer. Lecture Notes in Computer Science No. 1148; Proc. 1st ACM Workshop on Applied Computational Geometry (WACG), Federated Computing Research Conference 1996, Philadelphia, USA.

[3] J. C. Gyllenhaal, W. W. Hwu, and B. R. Rau. HMDES version 2.0 specification. Technical Report IMPACT-96-3, University of Illinois, Urbana Champagne, March 1996.

[4] Examples include the IBM PowerPC, the DEC Alpha, the HP PA-RISC, the SUN SPARC

[4] Real/Expr homepage, 1996. Source code, documentation, tutorial, examples and related literature available from the URL `http://simulation.nyu.edu/projects/exact/`.

[5] Trimaran homepage, 1998. A publicly released state-of-art system for research in compiler-optimization and architechure. Joint effort of Hewlett-Packard Laboratories, the IMPACT group of the University of Illinois and the ReaCT-ILP project of New York University. `http://www.trimaran.org/`.

[6] K. Ouchi. Real/Expr: Implementation of an exact computation package. Master's thesis, New York University, Department of Computer Science, Courant Institute, January 1997.

[7] D. A. Patterson and J. L. Hennessy. *Computer Architecture: a Quantitative Approach*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990. (with an appendix on Computer Arithmetic by David Goldberg).

[8] D. Richardson. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic*, 33(4), 1968.

[9] L. N. Trefethen and I. David Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[10] C. K. Yap. Robust geometric computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 35, pages 653–668. CRC Press LLC, 1997. 52 chapters, approx. xiv+987 pp.