High-Performance Decimal Floating-Point Arithmetic: Algorithms and Implementation in C++

MATTHEW BORLAND[*] and CHRISTOPHER KORMANYOS[*]

This article presents a comprehensive, portable C++ implementation of decimal floating-point arithmetic conforming to IEEE 754-2019 standards[4]. Our system offers three IEEE 754-compliant types, and three additional types that prioritize performance over strict standard adherence, providing flexible options for various computational needs. We describe in detail the Decimal system architecture, its standard library, and usage guidelines. The implementation incorporates novel algorithms for key operations, significantly improving performance in common use cases. Rigorous testing results demonstrate the system's correctness and IEEE 754 compliance. Performance benchmarks show competitive or superior results compared to existing implementations. This work addresses the growing demand for precise decimal arithmetic in financial, scientific, and engineering applications, offering a robust, efficient solution for C++ developers.

CCS Concepts: • **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

Additional Key Words and Phrases: Do, Not, Us, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:
Matthew Borland and Christopher Kormanyos. 2018. High-Performance Decimal Floating-Point Arithmetic: Algorithms and Implementation in C++. *ACM Trans. Math. Softw.* 37, 4, Article 111 (August 2018), 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Decimal floating-point arithmetic is crucial for many applications, particularly in financial and scientific computing. While several C++ decimal floating-point packages exist, they often lack IEEE 754 conformance, interoperability with the C++ Standard Template Library (STL), or both, and have limited portability. This paper presents a novel decimal system that addresses these limitations and advances decimal floating-point technology. Our system is standalone, relies on a minimal subset of the C++ STL, and has been tested on a wide range of devices, from S390X mainframes to AVR boards. It provides seamless interoperability with existing C++ standard types and full IEEE 754 conformance, features not collectively offered by any known package. This work significantly enhances the toolset available for high-precision decimal computations across diverse computing environments.

[*]Both authors contributed equally to this research.

Authors' address: Matthew Borland, matt@mattborland.com; Christopher Kormanyos, e_float@yahoo.com.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 0098-3500/2018/8-ART111
https://doi.org/XXXXXXX.XXXXXXX

2 THE DECIMAL SYSTEM

2.1 System Architecture

The decimal system architecture is robust and flexible. Each type is implemented completely independently of each other, but they share many of the same function implementations from the STL.

2.2 Decimal Types

The decimal system provides three IEEE-754 compliant types: `decimal32`, `decimal64`, and `decimal128`, as well as three analogous but non-conformant types: `decimal32_fast`, `decimal64_fast`, and `decimal128_fast`.

3 IMPLEMENTATION DETAILS

4 RESULTS

4.1 Testing and Precision

4.2 Performance

5 CONCLUSION AND OUTLOOK

ACKNOWLEDGMENTS

To the C++ Alliance for sponsoring the development of this library.

6 REFERENCES

REFERENCES

[1] Matt Borland, Peter Dimov, Junekey Jeon, Alexander Grund, Andrzej Krzemieński, Dmitry, Vinnie Falco, and Sam Darwin. 2024. *boostorg/charconv: Boost 1.86.0*. https://doi.org/10.5281/zenodo.13323694

[2] Michael F. Cowlishaw. 2003. Decimal Floating-Point: Algorism for Computers. In *Proceedings of the 16th IEEE Symposium on Computer Arithmetic*. IEEE, 104–111. https://doi.org/10.1109/ARITH.2003.1207670

[3] John F. Hart, E. W. Cheney, Charles L. Lawson, Hans J. Maehly, Charles K. Mesztenyi, John R. Rice, Henry G. Thacher, and Christoph Witzgall. 1968. *Computer Approximations*. John Wiley & Sons, New York.

[4] IEEE. 2019. *IEEE Standard for Floating-Point Arithmetic*. Standard IEEE 754-2019. IEEE Computer Society, New York, NY, USA. https://doi.org/10.1109/IEEESTD.2019.8766229

[5] Donald E. Knuth. 1997. *The Art of Computer Programming* (3rd ed.). Vol. 1-4B. Addison-Wesley, Reading, MA.

[6] Christopher Kormanyos. 2011. Algorithm 910: A Portable C++ Multiple-Precision System for Special-Function Calculations. *ACM Transactions on Mathematical Software (TOMS)* 37, 4, Article 45 (feb 2011), 27 pages. https://doi.org/10.1145/1916461.1916469

[7] Jean-Michel Muller. 2016. *Elementary Functions: Algorithms and Implementation* (3 ed.). Birkhäuser, Boston. https://doi.org/10.1007/978-1-4899-7983-4

[8] Jean-Michel Muller, Nicolas Brunie, Florent de Dinechin, Claude-Pierre Jeannerod, Mioara Joldes, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, and Serge Torres. 2018. *Handbook of Floating-Point Arithmetic* (2 ed.). Birkhäuser, Cham, Switzerland. https://doi.org/10.1007/978-3-319-76526-6

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009