
Preface to the Third Edition

Since the publication of the second edition of this book, in 2006, several important changes have occurred in the domain of computer arithmetic.

First, a new version of the IEEE-754 Standard for Floating-Point Arithmetic was adopted in June 2008. This new version was merged with the previous binary (754) and “radix independent” (854) standards, resolved some ambiguities of the previous release, standardized the fused multiply-add (FMA) instruction, and included new formats (among them, the binary128 format, previously called “quad precision”). An appendix to this new IEEE 754-2008 standard also makes some important recommendations concerning the elementary functions.

New tools have been released that make much easier the work of a programmer eager to implement very accurate functions. A typical example is Sollya.¹ Sollya offers, among many interesting features, a certified supremum norm of the difference between a polynomial and a function. It also computes very good polynomial approximations with constraints (such as requiring the coefficients to be exactly representable in a given format). Another example is Gappa,² which simplifies the calculation of error bounds for small “straight-line” numerical programs (such as those designed for evaluating elementary functions), and makes it possible to use proof checkers such as Coq for certifying these bounds. FloPoCo³ is a wonderful tool for implementing floating-point functions on FPGAs.

Research in this area is still very active. To cite a few examples: Harrison designed clever techniques for implementing decimal transcendental functions using the binary functions; Chevillard, Harrison, Joldes, and Lauter introduced a new algorithm for computing certified supremum norms of approximation errors; Johansson designed new algorithms for implementing functions in the “medium precision” range; Brunie et al. designed code generators for mathematical functions; several authors (especially de Dinechin) introduced hardware-oriented techniques targeted at FPGA implementation; Brisebarre and colleagues introduced methods for rigorous polynomial approximation.

¹ Available at <http://sollya.gforge.inria.fr>.

² Available at <http://gappa.gforge.inria.fr>.

³ Available at <http://flopoco.gforge.inria.fr>.

Acknowledgments

I have benefited much from discussion with my colleagues and students. Especially, I am grateful to Jean-Claude Bajard, Jean-Luc Beuchat, Sylvie Boldo, Nicolas Brisebarre, Laurent-Stéphane Didier, Florent de Dinechin, Miloš Ercegovic, Stef Graillat, Guillaume Hanrot, Claude-Pierre Jeannerod, Mioara Joldes, Peter Kornerup, Christoph Lauter, Vincent Lefèvre, Nicolas Louvet, Erik Martin-Dorel, Guillaume Melquiond, Marc Mezzarobba, Adrien Panhaleux, Antoine Plet, Valentina Popescu, Bruno Salvy, Peter Tang, Arnaud Tisserand, and Serge Torres.

As usual, working with Birkhäuser's staff on the publication of a book is a pleasure.

This third edition was prepared in LaTeX on an Apple MacBook Pro laptop. I used the book document class of LaTeX with a few modifications, and the Xfig drawing tool, the TikZ package, GNU-Plot, or Maple for most figures. The text editor I used is TeXShop.

Lyon
April 2016

Jean-Michel Muller

Preface to the Second Edition

Since the publication of the first edition of this book, many authors have introduced new techniques or improved existing ones. Examples are the *bipartite table method*, originally suggested by DasSarma and Matula in a seminal paper that led Schulte and Stine, and De Dinechin and Tisserand to design interesting improvements, the work on *formal proofs of floating-point algorithms* by (among others) John Harrison, David Russinoff, Laurent Théry, Marc Daumas and Sylvie Boldo, the design of *very accurate elementary function libraries* by people such as Peter Markstein, Shane Story, Peter Tang, David Defour and Florent de Dinechin, and the recently obtained results on the *table maker's dilemma* by Vincent Lefèvre. I therefore decided to present these new results in a new edition. Also, several colleagues and readers told me that a chapter devoted to multiple-precision arithmetic was missing in the previous edition. Chapter 7 now deals with that topic.

Computer arithmetic is changing rapidly. While I am writing these lines, the IEEE-754 Standard for Floating Point Arithmetic is being revised.⁴ Various technological evolutions have a deep impact on determining which algorithms are interesting and which are not. The complexity of the architecture of recent processors must be taken into account if we wish to design high-quality function software: we cannot ignore the notions of pipelining, memory cache and branch prediction and still write efficient software. Also, the possible availability of a fused multiply-accumulate instruction is an important parameter to consider when choosing an elementary function algorithm.

A detailed presentation of the contents is given in the introduction. After a preliminary chapter that presents a few notions on computer arithmetic, the book is divided into three major parts. The first part consists of three chapters and is devoted to algorithms using polynomial or rational approximations of the elementary functions and, possibly, tables. The last chapter of the first part deals with multiple-precision arithmetic. The second part consists of three chapters, and deals with “shift-and-add” algorithms, i.e., hardware-oriented algorithms that use additions and shifts only. The last part consists of four chapters. The first two chapters discuss issues that are

⁴For information, see <http://754r.ucbtest.org/>.

important when accuracy is a major goal (namely, range reduction, monotonicity and correct rounding). The third one mainly deals with exceptions. The last chapter gives some examples of implementation.

Acknowledgments

I would like to thank all those who suggested corrections and improvements to the first edition, or whose comments helped me to prepare this one. Discussions with Nick Higham, John Harrison and William Kahan have been enlightening. Shane Story, Paul Zimmermann, Vincent Lefèvre, Brian Shoemaker, Timm Ahrendt, Nelson H.F. Beebe, Tom Lynch suggested many corrections/modifications of the first edition. Nick Higham, Paul Zimmermann, Vincent Lefèvre, Florent de Dinechin, Sylvie Boldo, Nicolas Brisebarre, Miloš Ercegovac, Nathalie Revol read preliminary versions of this one. Working and conversing everyday with Jean-Luc Beuchat, Catherine Daramy, Marc Daumas, David Defour and Arnaud Tisserand has significantly deepened my knowledge of floating-point arithmetic and function calculation.

I owe a big “thank you” to Michel Cosnard, Miloš Ercegovac, Peter Kornerup and Tomas Lang. They helped me greatly when I was a young researcher, and with the passing years they have become good friends.

Working with Birkhäuser’s staff on the publication of this edition and the previous one has been a pleasure. I have been impressed by the quality of the help they provide their authors.

Since the writing of the first edition, my life has changed a lot: two wonderful daughters, Émilie and Camille, are now enlightening my existence. This book is dedicated to them and to my wife Marie Laure.

This second edition was typeset in LaTeX on a DELL laptop. I used the book document style with a few modifications, and the Xfig drawing tool or Maple for most figures. The text editor I used is the excellent WinEdt software, by Aleksander Simonic (see <http://www.winedt.com>).

Lyon
April 2005

Jean-Michel Muller

Preface to the First Edition

The elementary functions (sine, cosine, exponentials, logarithms...) are the most commonly used mathematical functions. Computing them quickly and accurately is a major goal in computer arithmetic. This book gives the theoretical background necessary to understand and/or build algorithms for computing these functions, presents algorithms (hardware-oriented as well as software-oriented), and discusses issues related to the accurate floating-point implementation of these functions. My purpose was not to give “cooking recipes” that allow to implement some given functions on some given floating-point systems, but to provide the reader with the knowledge that is necessary to build, or adapt algorithms to his or her computing environment.

When writing this book, I have had in mind two different audiences: *specialists*, who will have to design floating-point systems (hardware or software parts) or to do research on algorithms, and *inquiring minds*, who just want to know what kind of methods are used to compute the math functions in current computers or pocket calculators. Because of this, the book is intended to be helpful as well for postgraduate and advanced undergraduate students in computer science or applied mathematics as for professionals engaged in the design of algorithms, programs or circuits that implement floating-point arithmetic, or simply for engineers or scientists who want to improve their culture in that domain. Much of the book can be understood with only a basic grounding in computer science and mathematics: the basic notions on computer arithmetic that are necessary to understand are recalled in the first chapter.

The previous books on the same topic (mainly Hart et al. book *Computer Approximation* and Cody and Waite’s book *Software Manual for the Elementary Functions*) contained many coefficients of polynomial or rational approximations of the elementary functions. I have included relatively few such coefficients here, firstly to reduce the length of the book – since I also wanted to present the shift-and-add algorithms –, and secondly because today it is very easy to obtain them using Maple or a similar system: my primary concern is to explain how they can be computed and how they can be used. Moreover, the previous books on elementary functions essentially focused on *software* implementations and polynomial or rational approximations, whereas now these functions are frequently implemented (at least partially) in hardware, using different methods (table-based methods or shift-and-add algorithms, such as CORDIC): I have wanted to show a large spectrum

of methods. Whereas some years ago a library providing elementary functions with one or two incorrect bits only was considered adequate, current systems must be much more accurate. The next step will be to provide *correctly rounded* functions (at least for some functions, in some domains), i.e., the returned result should always be the “machine number” that is closest to the exact result. This goal has already been reached by some implementations in single precision. I try to show that it can be reached in higher precisions.

Acknowledgments

Many people helped me during the process of writing this book. Many others gave me, during enlightening conversations, some views on the problem that deeply influenced me. It is not possible to cite everybody, but among those persons, I would especially like to thank:

- Jean-Marc Delosme, Warren Ferguson, Tomas Lang, Steve Sommars, Naofumi Takagi, Roger Woods and Dan Zuras, who volunteered to read parts of this book and gave me good advice, and Charles Dunham, who provided me with interesting information;
- my former and current students Jean-Claude Bajard, Catherine Billet, Marc Daumas, Yvan Herreros, Sylvanus Kla, Vincent Lefèvre Christophe Mazenc, Xavier Merrheim (who invented the tale presented at the beginning of Chapter 8), Arnaud Tisserand and Hong-Jin Yeh;
- the staff of the Computer Science Department and LIP laboratory at ENS Lyon.

Working with Birkhäuser on the publication of this book was a pleasure.

And of course, I thank my wife, Marie Laure, to whom this book is dedicated, for her patience and help during the preparation of the manuscript.

This book was typeset in LaTeX on a SUN workstation and an Apple Macintosh. I used the book document style. The text editors I used are Keheler’s Alpha and GNU Emacs (Free Software Foundation).

Lyon
March 1997

Jean-Michel Muller

Elementary Functions

Algorithms and Implementation

Muller, J.-M.

2016, XXV, 283 p. 40 illus., Hardcover

ISBN: 978-1-4899-7981-0

A product of Birkhäuser Basel