

Cambridge University Press  
978-0-521-19469-3 - Modern Computer Arithmetic  
Richard P. Brent and Paul Zimmermann  
Frontmatter  
[More information](#)

**CAMBRIDGE MONOGRAPHS ON  
APPLIED AND COMPUTATIONAL  
MATHEMATICS**

---

**Series Editors**

M. ABLOWITZ, S. DAVIS, J. HINCH,  
A. ISERLES, J. OCKENDON, P. OLVER

---

**18      Modern Computer Arithmetic**

Cambridge University Press  
978-0-521-19469-3 - Modern Computer Arithmetic  
Richard P. Brent and Paul Zimmermann  
Frontmatter  
[More information](#)

The *Cambridge Monographs on Applied and Computational Mathematics* series reflects the crucial role of mathematical and computational techniques in contemporary science. The series publishes expositions on all aspects of applicable and numerical mathematics, with an emphasis on new developments in this fast-moving area of research.

State-of-the-art methods and algorithms as well as modern mathematical descriptions of physical and mechanical ideas are presented in a manner suited to graduate research students and professionals alike. Sound pedagogical presentation is a prerequisite. It is intended that books in the series will serve to inform a new generation of researchers.

A complete list of books in the series can be found at  
<http://www.cambridge.org/uk/series/sSeries.asp?code=MACM>  
Recent titles include the following:

6. The theory of composites, *Graeme W. Milton*
7. Geometry and topology for mesh generation, *Herbert Edelsbrunner*
8. Schwarz–Christoffel mapping, *Tobin A. Driscoll & Lloyd N. Trefethen*
9. High-order methods for incompressible fluid flow, *M. O. Deville, P. F. Fischer & E. H. Mund*
10. Practical extrapolation methods, *Avram Sidi*
11. Generalized Riemann problems in computational fluid dynamics, *Matania Ben-Artzi & Joseph Falcovitz*
12. Radial basis functions, *Martin D. Buhmann*
13. Iterative Krylov methods for large linear systems, *Henk van der Vorst*
14. Simulating Hamiltonian dynamics, *Benedict Leimkuhler & Sebastian Reich*
15. Collocation methods for Volterra integral and related functional differential equations, *Hermann Brunner*
16. Topology for computing, *Afra J. Zomorodian*
17. Scattered data approximation, *Holger Wendland*
18. Modern computer arithmetic, *Richard P. Brent & Paul Zimmermann*
19. Matrix preconditioning techniques and applications, *Ke Chen*
21. Spectral methods for time-dependent problems, *Jan Hesthaven, Sigal Gottlieb & David Gottlieb*
22. The mathematical foundations of mixing, *Rob Sturman, Julio M. Ottino & Stephen Wiggins*
23. Curve and surface reconstruction, *Tamal K. Dey*
24. Learning theory, *Felipe Cucker & Ding Xuan Zhou*
25. Algebraic geometry and statistical learning theory, *Sumio Watanabe*
26. A practical guide to the invariant calculus, *Elizabeth Louise Mansfield*

Cambridge University Press  
978-0-521-19469-3 - Modern Computer Arithmetic  
Richard P. Brent and Paul Zimmermann  
Frontmatter  
[More information](#)

---

# Modern Computer Arithmetic

RICHARD P. BRENT  
*Australian National University, Canberra*

PAUL ZIMMERMANN  
*INRIA, Nancy*



**CAMBRIDGE**  
UNIVERSITY PRESS

Cambridge University Press  
978-0-521-19469-3 - Modern Computer Arithmetic  
Richard P. Brent and Paul Zimmermann  
Frontmatter  
[More information](#)

---

CAMBRIDGE UNIVERSITY PRESS  
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore,  
São Paulo, Delhi, Dubai, Tokyo, Mexico City

Cambridge University Press  
The Edinburgh Building, Cambridge CB2 8RU, UK

Published in the United States of America by Cambridge University Press, New York

[www.cambridge.org](http://www.cambridge.org)  
Information on this title: [www.cambridge.org/9780521194693](http://www.cambridge.org/9780521194693)

© R. Brent and P. Zimmermann 2011

This publication is in copyright. Subject to statutory exception  
and to the provisions of relevant collective licensing agreements,  
no reproduction of any part may take place without the written  
permission of Cambridge University Press.

First published 2011

Printed in the United Kingdom at the University Press, Cambridge

*A catalogue record for this publication is available from the British Library*

*Library of Congress Cataloguing in Publication data*

ISBN 978-0-521-19469-3 Hardback

---

Cambridge University Press has no responsibility for the persistence or  
accuracy of URLs for external or third-party internet websites referred to  
in this publication, and does not guarantee that any content on such  
websites is, or will remain, accurate or appropriate.

---

## Contents

	<i>Preface</i>	<i>page</i> ix
	<i>Acknowledgements</i>	xi
	<i>Notation</i>	xiii
<b>1</b>	<b>Integer arithmetic</b>	1
	1.1 Representation and notations	1
	1.2 Addition and subtraction	2
	1.3 Multiplication	3
	1.3.1 Naive multiplication	4
	1.3.2 Karatsuba's algorithm	5
	1.3.3 Toom–Cook multiplication	6
	1.3.4 Use of the fast Fourier transform (FFT)	8
	1.3.5 Unbalanced multiplication	8
	1.3.6 Squaring	11
	1.3.7 Multiplication by a constant	13
	1.4 Division	14
	1.4.1 Naive division	14
	1.4.2 Divisor preconditioning	16
	1.4.3 Divide and conquer division	18
	1.4.4 Newton's method	21
	1.4.5 Exact division	21
	1.4.6 Only quotient or remainder wanted	22
	1.4.7 Division by a single word	23
	1.4.8 Hensel's division	24
	1.5 Roots	25
	1.5.1 Square root	25
	1.5.2 <i>k</i> th root	27
	1.5.3 Exact root	28

vi	<i>Contents</i>	
1.6	Greatest common divisor	29
1.6.1	Naive GCD	29
1.6.2	Extended GCD	32
1.6.3	Half binary GCD, divide and conquer GCD	33
1.7	Base conversion	37
1.7.1	Quadratic algorithms	37
1.7.2	Subquadratic algorithms	38
1.8	Exercises	39
1.9	Notes and references	44
<b>2</b>	<b>Modular arithmetic and the FFT</b>	<b>47</b>
2.1	Representation	47
2.1.1	Classical representation	47
2.1.2	Montgomery's form	48
2.1.3	Residue number systems	48
2.1.4	MSB vs LSB algorithms	49
2.1.5	Link with polynomials	49
2.2	Modular addition and subtraction	50
2.3	The Fourier transform	50
2.3.1	Theoretical setting	50
2.3.2	The fast Fourier transform	51
2.3.3	The Schönhage–Strassen algorithm	55
2.4	Modular multiplication	58
2.4.1	Barrett's algorithm	58
2.4.2	Montgomery's multiplication	60
2.4.3	McLaughlin's algorithm	63
2.4.4	Special moduli	65
2.5	Modular division and inversion	65
2.5.1	Several inversions at once	67
2.6	Modular exponentiation	68
2.6.1	Binary exponentiation	70
2.6.2	Exponentiation with a larger base	70
2.6.3	Sliding window and redundant representation	72
2.7	Chinese remainder theorem	73
2.8	Exercises	75
2.9	Notes and references	77
<b>3</b>	<b>Floating-point arithmetic</b>	<b>79</b>
3.1	Representation	79
3.1.1	Radix choice	80
3.1.2	Exponent range	81

<i>Contents</i>		vii
3.1.3	Special values	82
3.1.4	Subnormal numbers	82
3.1.5	Encoding	83
3.1.6	Precision: local, global, operation, operand	84
3.1.7	Link to integers	86
3.1.8	Ziv's algorithm and error analysis	86
3.1.9	Rounding	87
3.1.10	Strategies	90
3.2	Addition, subtraction, comparison	91
3.2.1	Floating-point addition	92
3.2.2	Floating-point subtraction	93
3.3	Multiplication	95
3.3.1	Integer multiplication via complex FFT	98
3.3.2	The middle product	99
3.4	Reciprocal and division	101
3.4.1	Reciprocal	102
3.4.2	Division	106
3.5	Square root	111
3.5.1	Reciprocal square root	112
3.6	Conversion	114
3.6.1	Floating-point output	115
3.6.2	Floating-point input	117
3.7	Exercises	118
3.8	Notes and references	120
<b>4</b>	<b>Elementary and special function evaluation</b>	<b>125</b>
4.1	Introduction	125
4.2	Newton's method	126
4.2.1	Newton's method for inverse roots	127
4.2.2	Newton's method for reciprocals	128
4.2.3	Newton's method for (reciprocal) square roots	129
4.2.4	Newton's method for formal power series	129
4.2.5	Newton's method for functional inverses	130
4.2.6	Higher-order Newton-like methods	131
4.3	Argument reduction	132
4.3.1	Repeated use of a doubling formula	134
4.3.2	Loss of precision	134
4.3.3	Guard digits	135
4.3.4	Doubling versus tripling	136
4.4	Power series	136

4.4.1	Direct power series evaluation	140
4.4.2	Power series with argument reduction	140
4.4.3	Rectangular series splitting	141
4.5	Asymptotic expansions	144
4.6	Continued fractions	150
4.7	Recurrence relations	152
4.7.1	Evaluation of Bessel functions	153
4.7.2	Evaluation of Bernoulli and tangent numbers	154
4.8	Arithmetic-geometric mean	158
4.8.1	Elliptic integrals	158
4.8.2	First AGM algorithm for the logarithm	159
4.8.3	Theta functions	160
4.8.4	Second AGM algorithm for the logarithm	162
4.8.5	The complex AGM	163
4.9	Binary splitting	163
4.9.1	A binary splitting algorithm for sin, cos	166
4.9.2	The bit-burst algorithm	167
4.10	Contour integration	169
4.11	Exercises	171
4.12	Notes and references	179
<b>5</b>	<b>Implementations and pointers</b>	<b>185</b>
5.1	Software tools	185
5.1.1	CLN	185
5.1.2	GNU MP (GMP)	185
5.1.3	MPFQ	186
5.1.4	GNU MPFR	187
5.1.5	Other multiple-precision packages	187
5.1.6	Computational algebra packages	188
5.2	Mailing lists	189
5.2.1	The GMP lists	189
5.2.2	The MPFR list	190
5.3	On-line documents	190
	<i>References</i>	191
	<i>Index</i>	207



## Preface

This is a book about algorithms for performing arithmetic, and their implementation on modern computers. We are concerned with software more than hardware – we do not cover computer architecture or the design of computer hardware since good books are already available on these topics. Instead, we focus on algorithms for efficiently performing arithmetic operations such as addition, multiplication, and division, and their connections to topics such as modular arithmetic, greatest common divisors, the fast Fourier transform (FFT), and the computation of special functions.

The algorithms that we present are mainly intended for arbitrary-precision arithmetic. That is, they are not limited by the computer wordsize of 32 or 64 bits, only by the memory and time available for the computation. We consider both integer and real (floating-point) computations.

The book is divided into four main chapters, plus one short chapter (essentially an appendix). Chapter 1 covers integer arithmetic. This has, of course, been considered in many other books and papers. However, there has been much recent progress, inspired in part by the application to public key cryptography, so most of the published books are now partly out of date or incomplete. Our aim is to present the latest developments in a concise manner. At the same time, we provide a self-contained introduction for the reader who is not an expert in the field.

Chapter 2 is concerned with modular arithmetic and the FFT, and their applications to computer arithmetic. We consider different number representations, fast algorithms for multiplication, division and exponentiation, and the use of the Chinese remainder theorem (CRT).

Chapter 3 covers floating-point arithmetic. Our concern is with high-precision floating-point arithmetic, implemented in software if the precision provided by the hardware (typically IEEE standard 53-bit significand) is

inadequate. The algorithms described in this chapter focus on *correct rounding*, extending the IEEE standard to arbitrary precision.

Chapter 4 deals with the computation, to arbitrary precision, of functions such as  $\sqrt{x}$ ,  $\exp$ ,  $\ln$ ,  $\sin$ ,  $\cos$ , and more generally functions defined by power series or continued fractions. Of course, the computation of special functions is a huge topic so we have had to be selective. In particular, we have concentrated on methods that are efficient and suitable for arbitrary-precision computations.

The last chapter contains pointers to implementations, useful web sites, mailing lists, and so on. Finally, at the end there is a one-page *Summary of complexities* which should be a useful *aide-mémoire*.

The chapters are fairly self-contained, so it is possible to read them out of order. For example, Chapter 4 could be read before Chapters 1–3, and Chapter 5 can be consulted at any time. Some topics, such as Newton's method, appear in different guises in several chapters. Cross-references are given where appropriate.

For details that are omitted, we give pointers in the *Notes and references* sections of each chapter, as well as in the bibliography. We have tried, as far as possible, to keep the main text uncluttered by footnotes and references, so most references are given in the Notes and references sections.

The book is intended for anyone interested in the design and implementation of efficient algorithms for computer arithmetic, and more generally efficient numerical algorithms. We did our best to present algorithms that are ready to implement in your favorite language, while keeping a high-level description and not getting too involved in low-level or machine-dependent details. An alphabetical list of algorithms can be found in the index.

Although the book is not specifically intended as a textbook, it could be used in a graduate course in mathematics or computer science, and for this reason, as well as to cover topics that could not be discussed at length in the text, we have included exercises at the end of each chapter. The exercises vary considerably in difficulty, from easy to small research projects, but we have not attempted to assign them a numerical rating. For solutions to the exercises, please contact the authors.

We welcome comments and corrections. Please send them to either of the authors.

Richard Brent and Paul Zimmermann  
Canberra and Nancy  
MCA@rpbrent.com  
Paul.Zimmermann@inria.fr

## Acknowledgements

We thank the French National Institute for Research in Computer Science and Control (INRIA), the Australian National University (ANU), and the Australian Research Council (ARC), for their support. The book could not have been written without the contributions of many friends and colleagues, too numerous to mention here, but acknowledged in the text and in the Notes and references sections at the end of each chapter.

We also thank those who have sent us comments on and corrections to earlier versions of this book: Jörg Arndt, Marco Bodrato, Wolfgang Ehrhardt (with special thanks), Steven Galbraith, Torbjörn Granlund, Guillaume Hanrot, Marc Mezzarobba, Jean-Michel Muller, Denis Roegel, Wolfgang Schmid, Arnold Schönhage, Sidi Mohamed Sedjelmaci, Emmanuel Thomé, and Mark Wezelenburg. Two anonymous reviewers provided very helpful suggestions. Jérémie Detrey and Anne Rix helped us in the copy-editing phase.

The *Mathematics Genealogy Project* (<http://www.genealogy.ams.org/>) and Don Knuth's *The Art of Computer Programming* [142] were useful resources for details of entries in the index.

We also thank the authors of the  $\text{\LaTeX}$  program, which allowed us to produce this book, the authors of the `gnuplot` program, and the authors of the GNU MP library, which helped us to illustrate several algorithms with concrete figures.

Finally, we acknowledge the contribution of Erin Brent, who first suggested writing the book; and thank our wives, Judy-anne and Marie, for their patience and encouragement.

## Notation

$\mathbb{C}$	set of complex numbers
$\widehat{\mathbb{C}}$	set of extended complex numbers $\mathbb{C} \cup \{\infty\}$
$\mathbb{N}$	set of natural numbers (nonnegative integers)
$\mathbb{N}^*$	set of positive integers $\mathbb{N} \setminus \{0\}$
$\mathbb{Q}$	set of rational numbers
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integers
$\mathbb{Z}/n\mathbb{Z}$	ring of residues modulo $n$
$C^n$	set of (real or complex) functions with $n$ continuous derivatives in the region of interest
$\Re(z)$	real part of a complex number $z$
$\Im(z)$	imaginary part of a complex number $z$
$\bar{z}$	conjugate of a complex number $z$
$ z $	Euclidean norm of a complex number $z$ , or absolute value of a scalar $z$
$B_n$	Bernoulli numbers, $\sum_{n \geq 0} B_n z^n / n! = z / (e^z - 1)$
$C_n$	scaled Bernoulli numbers, $C_n = B_{2n} / (2n)!$ , $\sum C_n z^{2n} = (z/2) / \tanh(z/2)$
$T_n$	tangent numbers, $\sum T_n z^{2n-1} / (2n-1)! = \tan z$
$H_n$	harmonic number $\sum_{j=1}^n 1/j$ (0 if $n \leq 0$ )
$\binom{n}{k}$	binomial coefficient “ $n$ choose $k$ ” = $n! / (k! (n-k)!)$ (0 if $k < 0$ or $k > n$ )

xiv	<i>Notation</i>
$\beta$	“word” base (usually $2^{32}$ or $2^{64}$ ) or “radix” (floating-point)
$n$	“precision”: number of base $\beta$ digits in an integer or in a floating-point significand, or a free variable
$\varepsilon$	“machine precision” $\beta^{1-n}/2$ or (in complexity bounds) an arbitrarily small positive constant
$\eta$	smallest positive subnormal number
$\circ(x), \circ_n(x)$	rounding of real number $x$ in precision $n$ (Definition 3.1)
$\text{ulp}(x)$	for a floating-point number $x$ , one unit in the last place
$M(n)$	time to multiply $n$ -bit integers, or polynomials of degree $n - 1$ , depending on the context
$\sim M(n)$	a function $f(n)$ such that $f(n)/M(n) \rightarrow 1$ as $n \rightarrow \infty$ (we sometimes lazily omit the “ $\sim$ ” if the meaning is clear)
$M(m, n)$	time to multiply an $m$ -bit integer by an $n$ -bit integer
$D(n)$	time to divide a $2n$ -bit integer by an $n$ -bit integer, giving quotient and remainder
$D(m, n)$	time to divide an $m$ -bit integer by an $n$ -bit integer, giving quotient and remainder
$a b$	$a$ is a divisor of $b$ , that is $b = ka$ for some $k \in \mathbb{Z}$
$a = b \bmod m$	modular equality, $m (a - b)$
$q \leftarrow a \text{ div } b$	assignment of integer quotient to $q$ ( $0 \leq a - qb < b$ )
$r \leftarrow a \bmod b$	assignment of integer remainder to $r$ ( $0 \leq r = a - qb < b$ )
$(a, b)$	greatest common divisor of $a$ and $b$
$\left(\frac{a}{b}\right)$ or $(a b)$	Jacobi symbol ( $b$ odd and positive)
iff	if and only if
$i \wedge j$	bitwise <i>and</i> of integers $i$ and $j$ , or logical <i>and</i> of two Boolean expressions
$i \vee j$	bitwise <i>or</i> of integers $i$ and $j$ , or logical <i>or</i> of two Boolean expressions
$i \oplus j$	bitwise <i>exclusive-or</i> of integers $i$ and $j$
$i \ll k$	integer $i$ multiplied by $2^k$
$i \gg k$	quotient of division of integer $i$ by $2^k$
$a \cdot b, a \times b$	product of scalars $a, b$
$a * b$	cyclic convolution of vectors $a, b$
$\nu(n)$	2-valuation: largest $k$ such that $2^k$ divides $n$ ( $\nu(0) = \infty$ )
$\sigma(e)$	length of the shortest addition chain to compute $e$
$\phi(n)$	Euler’s totient function, $\#\{m : 0 < m \leq n \wedge (m, n) = 1\}$

Notation

$\deg(A)$	for a polynomial $A$ , the degree of $A$
$\text{ord}(A)$	for a power series $A = \sum_j a_j z^j$ , $\text{ord}(A) = \min\{j : a_j \neq 0\}$ ( $\text{ord}(0) = +\infty$ )
$\exp(x)$ or $e^x$	exponential function
$\ln(x)$	natural logarithm
$\log_b(x)$	base- $b$ logarithm $\ln(x)/\ln(b)$
$\lg(x)$	base-2 logarithm $\ln(x)/\ln(2) = \log_2(x)$
$\log(x)$	logarithm to any fixed base
$\log^k(x)$	$(\log x)^k$
$\lceil x \rceil$	ceiling function, $\min\{n \in \mathbb{Z} : n \geq x\}$
$\lfloor x \rfloor$	floor function, $\max\{n \in \mathbb{Z} : n \leq x\}$
$[x]$	nearest integer function, $\lfloor x + 1/2 \rfloor$
$\text{sign}(n)$	+1 if $n > 0$ , -1 if $n < 0$ , and 0 if $n = 0$
$\text{nbits}(n)$	$\lfloor \lg(n) \rfloor + 1$ if $n > 0$ , 0 if $n = 0$
$[a, b]$	closed interval $\{x \in \mathbb{R} : a \leq x \leq b\}$ (empty if $a > b$ )
$(a, b)$	open interval $\{x \in \mathbb{R} : a < x < b\}$ (empty if $a \geq b$ )
$[a, b), (a, b]$	half-open intervals, $a \leq x < b$ , $a < x \leq b$ respectively
${}^t[a, b]$ or $[a, b]^t$	column vector $\begin{pmatrix} a \\ b \end{pmatrix}$
$[a, b; c, d]$	$2 \times 2$ matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$
$\hat{a}_j$	element of the (forward) Fourier transform of vector $a$
$\tilde{a}_j$	element of the backward Fourier transform of vector $a$
$f(n) = O(g(n))$	$\exists c, n_0$ such that $ f(n)  \leq cg(n)$ for all $n \geq n_0$
$f(n) = \Omega(g(n))$	$\exists c > 0, n_0$ such that $ f(n)  \geq cg(n)$ for all $n \geq n_0$
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$
$f(n) \sim g(n)$	$f(n)/g(n) \rightarrow 1$ as $n \rightarrow \infty$
$f(n) = o(g(n))$	$f(n)/g(n) \rightarrow 0$ as $n \rightarrow \infty$
$f(n) \ll g(n)$	$f(n) = O(g(n))$
$f(n) \gg g(n)$	$g(n) \ll f(n)$
$f(x) \sim \sum_0^n a_j/x^j$	$f(x) - \sum_0^n a_j/x^j = o(1/x^n)$ as $x \rightarrow +\infty$
123 456 789	123456789 (for large integers, we may use a space after every third digit)

xvi

*Notation*

$xxx.yyy_\rho$	a number $xxx.yyy$ written in base $\rho$ ; for example, the decimal number 3.25 is $11.01_2$ in binary
$\frac{a}{b+} \frac{c}{d+} \frac{e}{f+} \dots$	continued fraction $a/(b + c/(d + e/(f + \dots)))$
$ A $	determinant of a matrix $A$ , e.g. $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$
$\text{PV} \int_a^b f(x) dx$	Cauchy principal value integral, defined by a limit if $f$ has a singularity in $(a, b)$
$s    t$	concatenation of strings $s$ and $t$
$\triangleright \langle \text{text} \rangle$	comment in an algorithm
$\square$	end of a proof