

---

# Contents

<b>1</b>	<b>Introduction . . . . .</b>	<b>1</b>
<b>2</b>	<b>Introduction to Computer Arithmetic . . . . .</b>	<b>7</b>
2.1	Basic Notions of Floating-Point Arithmetic . . . . .	7
2.1.1	Basic Notions . . . . .	7
2.1.2	Rounding Functions. . . . .	9
2.1.3	ULPs. . . . .	12
2.1.4	Infinitely Precise Significand . . . . .	13
2.1.5	Fused Multiply–Add Operations . . . . .	14
2.1.6	The Formats Specified by the IEEE-754-2008 Standard for Floating-Point Arithmetic. . . . .	15
2.1.7	Testing Your Computational Environment . . . . .	16
2.2	Advanced Manipulation of FP Numbers. . . . .	17
2.2.1	Error-Free Transforms: Computing the Error of a FP Addition or Multiplication. . . . .	17
2.2.2	Manipulating Double-Word or Triple-Word Numbers . . . . .	18
2.2.3	An Example that Illustrates What We Have Learnt so Far . . . . .	21
2.2.4	The GAPPA Tool . . . . .	24
2.2.5	Maple Programs that Compute binary32 and binary64 Approximations . . . . .	27
2.2.6	The Future of Floating-Point Arithmetic. . . . .	29
2.3	Redundant Number Systems . . . . .	30
2.3.1	Signed-Digit Number Systems . . . . .	30
2.3.2	The Carry-Save and Borrow-Save Number Systems . . . . .	32
2.3.3	Canonical Recoding. . . . .	33
 <b>Part I Algorithms Based on Polynomial Approximation and/or Table Lookup, Multiple-Precision Evaluation of Functions</b>		
<b>3</b>	<b>The Classical Theory of Polynomial or Rational Approximations . . . . .</b>	<b>39</b>
3.1	What About Interpolation? . . . . .	40
3.2	Least Squares Polynomial Approximations . . . . .	41
3.2.1	Legendre Polynomials . . . . .	42
3.2.2	Chebyshev Polynomials . . . . .	42

3.2.3	Jacobi Polynomials . . . . .	44
3.2.4	Laguerre Polynomials . . . . .	44
3.2.5	Using These Orthogonal Polynomials in Any Interval . . . . .	44
3.3	Least Maximum Polynomial Approximations . . . . .	45
3.4	Some Examples . . . . .	46
3.5	Speed of Convergence . . . . .	52
3.6	Remez's Algorithm . . . . .	52
3.7	Minimizing the Maximum <i>Relative</i> Error . . . . .	57
3.8	Rational Approximations . . . . .	58
3.9	Accurately Computing Supremum Norms . . . . .	61
3.10	Actual Computation of Approximations . . . . .	63
<b>4</b>	<b>Polynomial Approximations with Special Constraints . . . . .</b>	<b>67</b>
4.1	Polynomials with Exactly Representable Coefficients . . . . .	71
4.1.1	An Iterative Method . . . . .	71
4.1.2	An Exact Method (For Small Degrees) . . . . .	72
4.1.3	A Method Based on Lattice-Reduction . . . . .	73
4.2	Getting Nearly Best Approximations Using Sollya . . . . .	75
4.3	Miscellaneous . . . . .	79
<b>5</b>	<b>Polynomial Evaluation . . . . .</b>	<b>81</b>
5.1	Sequential Evaluation of Polynomials . . . . .	81
5.1.1	Horner's Scheme . . . . .	81
5.1.2	Preprocessing of the Coefficients . . . . .	82
5.2	Evaluating Polynomials When Some Parallelism is Available . . . . .	84
5.2.1	Generalizations of Horner's Scheme . . . . .	84
5.2.2	Estrin's Method . . . . .	84
5.2.3	Evaluating Polynomials on Modern Processors . . . . .	85
5.3	Computing Bounds on the Evaluation Error . . . . .	87
5.3.1	Evaluation Error Assuming Horner's Scheme is Used . . . . .	88
5.3.2	Evaluation Error with Methods Different than Horner's Scheme . . . . .	96
5.3.3	When High Accuracy is Needed . . . . .	97
5.4	Polynomial Evaluation by Specific Hardware . . . . .	98
5.4.1	The E-Method . . . . .	98
5.4.2	Custom Precision Function Evaluation on Embedded Processors . . . . .	100
5.4.3	Polynomial Evaluation on FPGAs . . . . .	100
<b>6</b>	<b>Table-Based Methods . . . . .</b>	<b>101</b>
6.1	Introduction . . . . .	101
6.2	Table-Driven Algorithms . . . . .	103
6.2.1	Tang's Algorithm for $\exp(x)$ in IEEE Floating-Point Arithmetic . . . . .	104
6.2.2	$\ln(x)$ on $[1, 2]$ . . . . .	105
6.2.3	$\sin(x)$ on $[0, \pi/4]$ . . . . .	106

6.3	Gal's Accurate Tables Method . . . . .	107
6.4	Use of Pythagorean Triples . . . . .	110
6.5	Table Methods Requiring Specialized Hardware . . . . .	111
6.5.1	Wong and Goto's Algorithm for Computing Logarithms . . . . .	111
6.5.2	Wong and Goto's Algorithm for Computing Exponentials . . . . .	114
6.5.3	Ercegovac et al.'s Algorithm . . . . .	115
6.5.4	Bipartite and Multipartite Methods . . . . .	117
6.6	$(M, p, k)$ -Friendly Points: A Method Dedicated to Trigonometric Functions . . . . .	119
<b>7</b>	<b>Multiple-Precision Evaluation of Functions . . . . .</b>	<b>121</b>
7.1	Introduction . . . . .	121
7.2	Just a Few Words on Multiple-Precision Multiplication . . . . .	122
7.2.1	Karatsuba's Method . . . . .	122
7.2.2	The Toom-Cook Family of Multiplication Algorithms . . . . .	123
7.2.3	FFT-Based Methods . . . . .	126
7.3	Multiple-Precision Division and Square-Root . . . . .	126
7.3.1	The Newton–Raphson Iteration . . . . .	126
7.4	Algorithms Based on the Evaluation of Power Series . . . . .	128
7.4.1	Binary Splitting Techniques . . . . .	129
7.5	The Arithmetic–Geometric Mean (AGM) . . . . .	130
7.5.1	Presentation of the AGM . . . . .	130
7.5.2	Computing Logarithms with the AGM . . . . .	131
7.5.3	Computing Exponentials with the AGM . . . . .	133
7.5.4	Very Fast Computation of Trigonometric Functions . . . . .	133

## Part II Shift-and-Add Algorithms

<b>8</b>	<b>Introduction to Shift-and-Add Algorithms . . . . .</b>	<b>139</b>
8.1	The Restoring and Nonrestoring Algorithms . . . . .	140
8.2	Simple Algorithms for Exponentials and Logarithms . . . . .	145
8.2.1	The Restoring Algorithm for Exponentials . . . . .	145
8.2.2	The Restoring Algorithm for Logarithms . . . . .	146
8.3	Faster Shift-and-Add Algorithms . . . . .	147
8.3.1	Faster Computation of Exponentials . . . . .	147
8.3.2	Faster Computation of Logarithms . . . . .	152
8.4	Baker's Predictive Algorithm . . . . .	155
8.5	Bibliographic Notes . . . . .	163
<b>9</b>	<b>The CORDIC Algorithm . . . . .</b>	<b>165</b>
9.1	Introduction . . . . .	165
9.2	The Conventional CORDIC Iteration . . . . .	165
9.3	Acceleration of the Last Iterations . . . . .	170
9.4	Scale Factor Compensation . . . . .	170

9.5	CORDIC With Redundant Number Systems and a Variable Factor . . . . .	172
9.5.1	Signed-Digit Implementation. . . . .	173
9.5.2	Carry-Save Implementation. . . . .	174
9.5.3	The Variable Scale Factor Problem . . . . .	174
9.6	The Double Rotation Method . . . . .	174
9.7	The Branching CORDIC Algorithm. . . . .	177
9.8	The Differential CORDIC Algorithm. . . . .	177
9.9	The “CORDIC II” Approach . . . . .	180
9.10	Computation of $\cos^{-1}$ and $\sin^{-1}$ Using CORDIC . . . . .	181
9.11	Variations on CORDIC . . . . .	183
<b>10</b>	<b>Some Other Shift-and-Add Algorithms . . . . .</b>	<b>185</b>
10.1	High-Radix Algorithms . . . . .	185
10.1.1	Ercegovac’s Radix-16 Algorithms . . . . .	185
10.2	The BKM Algorithm. . . . .	189
10.2.1	The BKM Iteration . . . . .	189
10.2.2	Computation of the Exponential Function (E-mode) . . . . .	190
10.2.3	Computation of the Logarithm Function (L-mode) . . . . .	193
10.2.4	Application to the Computation of Elementary Functions. . . . .	194
 <b>Part III Range Reduction, Final Rounding and Exceptions</b>		
<b>11</b>	<b>Range Reduction . . . . .</b>	<b>199</b>
11.1	Introduction . . . . .	199
11.2	Cody and Waite’s Method for Range Reduction . . . . .	203
11.2.1	The Classical Cody–Waite Reduction. . . . .	203
11.2.2	When a Fused Multiply-add (FMA) Instruction is Available . . . . .	204
11.3	Finding Worst Cases for Range Reduction? . . . . .	206
11.3.1	A Few Basic Notions On Continued Fractions . . . . .	206
11.3.2	Finding Worst Cases Using Continued Fractions . . . . .	208
11.4	The Payne and Hanek Reduction Algorithm. . . . .	211
11.5	Modular Range Reduction Algorithms . . . . .	213
11.5.1	The MRR Algorithm . . . . .	213
11.5.2	The Double Residue Modular Range Reduction (DRMRR) Algorithm . . . . .	216
11.5.3	High Radix Modular Reduction. . . . .	217
<b>12</b>	<b>Final Rounding. . . . .</b>	<b>219</b>
12.1	Introduction . . . . .	219
12.2	Monotonicity . . . . .	220
12.3	Correct Rounding: Presentation of the Problem. . . . .	221
12.4	Ziv’s Rounding Test . . . . .	224
12.5	Some Experiments . . . . .	225
12.6	A “Probabilistic” Approach to the Problem . . . . .	226

12.7	Upper Bounds on $m$ . . . . .	229
12.7.1	Algebraic Functions. . . . .	229
12.7.2	Transcendental Functions . . . . .	229
12.8	Solving the TMD in Practice . . . . .	232
12.8.1	Special Input Values . . . . .	232
12.8.2	The L-Algorithm. . . . .	232
12.8.3	The SLZ Algorithm. . . . .	232
12.8.4	Nontrivial Hardest-to-Round Points Found So Far . . . . .	233
<b>13</b>	<b>Miscellaneous</b> . . . . .	245
13.1	Exceptions . . . . .	245
13.1.1	NaNs. . . . .	246
13.1.2	Exact Results . . . . .	247
13.2	Notes on $x^y$ . . . . .	248
13.3	Special Functions, Functions of Complex Numbers . . . .	250
<b>14</b>	<b>Examples of Implementation</b> . . . . .	253
14.1	First Example: The Cyrix FastMath Processor. . . . .	253
14.2	The INTEL Functions Designed for the Itanium Processor . . . . .	254
14.2.1	Sine and Cosine . . . . .	255
14.2.2	Arctangent . . . . .	255
14.3	The LIBULTIM Library. . . . .	256
14.4	The CRLIBM Library . . . . .	257
14.4.1	Computation of $\sin(x)$ or $\cos(x)$ (Quick Step). . . . .	257
14.4.2	Computation of $\ln(x)$ . . . . .	258
14.5	SUN's Former LIBMCR Library. . . . .	260
14.6	The HP-UX Compiler for the Itanium Processor . . . . .	260
14.7	The METALIBM Project . . . . .	261
	<b>Bibliography</b> . . . . .	263
	<b>Index</b> . . . . .	279



<http://www.springer.com/978-1-4899-7981-0>

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