

Contents

List of figures	xxv
List of tables	xxxi
Quick start	xxxv
1 Introduction	1
1.1 Programming conventions	2
1.2 Naming conventions	4
1.3 Library contributions and coverage	5
1.4 Summary	6
2 Iterative solutions and other tools	7
2.1 Polynomials and Taylor series	7
2.2 First-order Taylor series approximation	8
2.3 Second-order Taylor series approximation	9
2.4 Another second-order Taylor series approximation	9
2.5 Convergence of second-order methods	10
2.6 Taylor series for elementary functions	10
2.7 Continued fractions	12
2.8 Summation of continued fractions	17
2.9 Asymptotic expansions	19
2.10 Series inversion	20
2.11 Summary	22
3 Polynomial approximations	23
3.1 Computation of odd series	23
3.2 Computation of even series	25
3.3 Computation of general series	25
3.4 Limitations of Cody/Waite polynomials	28
3.5 Polynomial fits with Maple	32
3.6 Polynomial fits with Mathematica	33
3.7 Exact polynomial coefficients	42
3.8 Cody/Waite rational polynomials	43
3.9 Chebyshev polynomial economization	43
3.10 Evaluating Chebyshev polynomials	48
3.11 Error compensation in Chebyshev fits	50
3.12 Improving Chebyshev fits	51
3.13 Chebyshev fits in rational form	52
3.14 Chebyshev fits with Mathematica	56
3.15 Chebyshev fits for function representation	57
3.16 Extending the library	57
3.17 Summary and further reading	58

4	Implementation issues	61
4.1	Error magnification	61
4.2	Machine representation and machine epsilon	62
4.3	IEEE 754 arithmetic	63
4.4	Evaluation order in C	64
4.5	The <code>volatile</code> type qualifier	65
4.6	Rounding in floating-point arithmetic	66
4.7	Signed zero	69
4.7.1	Detecting the sign of zero	69
4.7.2	Signed-zero constants	69
4.7.3	Arc tangent and signed zero	70
4.8	Floating-point zero divide	70
4.9	Floating-point overflow	71
4.10	Integer overflow	72
4.10.1	Preventing integer overflow	74
4.10.1.1	Safe integer absolute value	74
4.10.1.2	Safe integer addition	75
4.10.1.3	Safe integer division	75
4.10.1.4	Safe integer multiplication	75
4.10.1.5	Safe integer negation	76
4.10.1.6	Safe integer remainder	76
4.10.1.7	Safe integer subtraction	76
4.10.1.8	Safe integer operations: a retrospective	77
4.11	Floating-point underflow	77
4.12	Subnormal numbers	78
4.13	Floating-point inexact operation	79
4.14	Floating-point invalid operation	79
4.15	Remarks on NaN tests	80
4.16	Ulp — units in the last place	81
4.17	Fused multiply-add	85
4.18	Fused multiply-add and polynomials	88
4.19	Significance loss	89
4.20	Error handling and reporting	89
4.21	Interpreting error codes	93
4.22	C99 changes to error reporting	94
4.23	Error reporting with threads	95
4.24	Comments on error reporting	95
4.25	Testing function implementations	96
4.25.1	Taylor-series evaluation	97
4.25.2	Argument purification	97
4.25.3	Addition-rule evaluation	98
4.25.4	Relative error computation	99
4.26	Extended data types on Hewlett–Packard HP-UX IA-64	100
4.27	Extensions for decimal arithmetic	101
4.28	Further reading	103
4.29	Summary	104
5	The floating-point environment	105
5.1	IEEE 754 and programming languages	105
5.2	IEEE 754 and the <code>mathcw</code> library	106
5.3	Exceptions and traps	106
5.4	Access to exception flags and rounding control	107
5.5	The environment access pragma	110
5.6	Implementation of exception-flag and rounding-control access	110

5.6.1	Clearing exception flags: <code>feclearexcept()</code>	110
5.6.2	Getting the rounding direction: <code>fegetround()</code>	111
5.6.3	Raising exception flags: <code>feraiseexcept()</code>	111
5.6.4	Setting the rounding direction: <code>fesetround()</code>	111
5.6.5	Testing exception flags: <code>fetestexcept()</code>	112
5.6.6	Comments on the core five	112
5.7	Using exception flags: simple cases	112
5.8	Using rounding control	115
5.9	Additional exception flag access	116
5.9.1	Getting the environment: <code>fegetenv()</code>	117
5.9.2	Setting the environment: <code>fesetenv()</code>	117
5.9.3	Getting exception flags: <code>fegetexceptflag()</code>	118
5.9.4	Setting exception flags: <code>fesetexceptflag()</code>	118
5.9.5	Holding exception flags: <code>feholdexcept()</code>	119
5.9.6	Updating the environment: <code>feupdateenv()</code>	119
5.9.7	Comments on the six functions	120
5.10	Using exception flags: complex case	120
5.11	Access to precision control	123
5.11.1	Precision control in hardware	124
5.11.2	Precision control and the AMD64 architecture	124
5.11.3	Precision control in the <code>mathcw</code> library	124
5.12	Using precision control	126
5.13	Summary	127
6	Converting floating-point values to integers	129
6.1	Integer conversion in programming languages	129
6.2	Programming issues for conversions to integers	130
6.3	Hardware out-of-range conversions	131
6.4	Rounding modes and integer conversions	132
6.5	Extracting integral and fractional parts	132
6.6	Truncation functions	135
6.7	Ceiling and floor functions	136
6.8	Floating-point rounding functions with fixed rounding	137
6.9	Floating-point rounding functions: current rounding	138
6.10	Floating-point rounding functions without <i>inexact</i> exception	139
6.11	Integer rounding functions with fixed rounding	140
6.12	Integer rounding functions with current rounding	142
6.13	Remainder	143
6.14	Why the remainder functions are hard	144
6.15	Computing <code>fmod()</code>	146
6.16	Computing <code>remainder()</code>	148
6.17	Computing <code>remquo()</code>	150
6.18	Computing one remainder from the other	152
6.19	Computing the remainder in nonbinary bases	155
6.20	Summary	156
7	Random numbers	157
7.1	Guidelines for random-number software	157
7.2	Creating generator seeds	158
7.3	Random floating-point values	160
7.4	Random integers from floating-point generator	165
7.5	Random integers from an integer generator	166
7.6	Random integers in ascending order	168
7.7	How random numbers are generated	169
7.7.1	Linear congruential generators	169

7.7.2	Deficiencies of congruential generators	170
7.7.3	Computing congruential generators	171
7.7.4	Faster congruential generators	174
7.7.5	Other generator algorithms	176
7.7.6	Combined generators	177
7.7.7	Cryptographic generators	178
7.8	Removing generator bias	178
7.9	Improving a poor random number generator	178
7.10	Why long periods matter	179
7.11	Inversive congruential generators	180
7.11.1	Digression: Euclid's algorithm	180
7.11.1.1	Euclid's algorithm for any integers	183
7.11.1.2	Division-free gcd algorithm	184
7.11.2	Another digression: the extended Euclid's algorithm	186
7.12	Inversive congruential generators, revisited	189
7.13	Distributions of random numbers	189
7.13.1	The uniform distribution	189
7.13.2	The exponential distribution	189
7.13.3	The logarithmic distribution	190
7.13.4	The normal distribution	192
7.13.5	More on the normal distribution	194
7.14	Other distributions	195
7.14.1	Numerical demonstration of the Central Limit Theorem	196
7.15	Testing random-number generators	196
7.15.1	The chi-square test	197
7.15.2	Random-number generator test suites	200
7.15.3	Testing generators for nonuniform distributions	202
7.16	Applications of random numbers	202
7.16.1	Monte Carlo quadrature	202
7.16.2	Encryption and decryption	203
7.16.2.1	Problems with cryptography	203
7.16.2.2	A provably secure encryption method	204
7.16.2.3	Demonstration of a one-time pad	204
7.16.2.4	Attacks on one-time-pad encryption	207
7.16.2.5	Choice of keys and encryption methods	207
7.16.2.6	Caveats about cryptography	208
7.17	The mathcw random number routines	208
7.18	Summary, advice, and further reading	214
8	Roots	215
8.1	Square root	215
8.1.1	Considerations for rounding of the square root	216
8.1.2	An algorithm for correct rounding of the square root	217
8.1.3	Variant iterations for the square root	220
8.2	Hypotenuse and vector norms	222
8.3	Hypotenuse by iteration	227
8.4	Reciprocal square root	233
8.4.1	Improved rounding of the reciprocal square root	234
8.4.2	Almost-correct rounding of the reciprocal square root	235
8.5	Cube root	237
8.5.1	Improved rounding of the cube root	237
8.5.2	Almost-correct rounding of the cube root	239
8.6	Roots in hardware	240
8.7	Summary	242

9	Argument reduction	243
9.1	Simple argument reduction	243
9.2	Exact argument reduction	250
9.3	Implementing exact argument reduction	253
9.4	Testing argument reduction	265
9.5	Retrospective on argument reduction	265
10	Exponential and logarithm	267
10.1	Exponential functions	267
10.2	Exponential near zero	273
10.3	Logarithm functions	282
10.3.1	Computing logarithms in a binary base	284
10.3.2	Computing logarithms in a decimal base	287
10.4	Logarithm near one	290
10.5	Exponential and logarithm in hardware	292
10.6	Compound interest and annuities	294
10.7	Summary	298
11	Trigonometric functions	299
11.1	Sine and cosine properties	299
11.2	Tangent properties	302
11.3	Argument conventions and units	304
11.4	Computing the cosine and sine	306
11.5	Computing the tangent	310
11.6	Trigonometric functions in degrees	313
11.7	Trigonometric functions in units of π	315
11.7.1	Cosine and sine in units of π	316
11.7.2	Cotangent and tangent in units of π	318
11.8	Computing the cosine and sine together	320
11.9	Inverse sine and cosine	323
11.10	Inverse tangent	331
11.11	Inverse tangent, take two	336
11.12	Trigonometric functions in hardware	338
11.13	Testing trigonometric functions	339
11.14	Retrospective on trigonometric functions	340
12	Hyperbolic functions	341
12.1	Hyperbolic functions	341
12.2	Improving the hyperbolic functions	345
12.3	Computing the hyperbolic functions together	348
12.4	Inverse hyperbolic functions	348
12.5	Hyperbolic functions in hardware	350
12.6	Summary	352
13	Pair-precision arithmetic	353
13.1	Limitations of pair-precision arithmetic	354
13.2	Design of the pair-precision software interface	355
13.3	Pair-precision initialization	356
13.4	Pair-precision evaluation	357
13.5	Pair-precision high part	357
13.6	Pair-precision low part	357
13.7	Pair-precision copy	357
13.8	Pair-precision negation	358
13.9	Pair-precision absolute value	358
13.10	Pair-precision sum	358

13.11	Splitting numbers into pair sums	359
13.12	Premature overflow in splitting	362
13.13	Pair-precision addition	365
13.14	Pair-precision subtraction	367
13.15	Pair-precision comparison	368
13.16	Pair-precision multiplication	368
13.17	Pair-precision division	371
13.18	Pair-precision square root	373
13.19	Pair-precision cube root	377
13.20	Accuracy of pair-precision arithmetic	379
13.21	Pair-precision vector sum	384
13.22	Exact vector sums	385
13.23	Pair-precision dot product	385
13.24	Pair-precision product sum	386
13.25	Pair-precision decimal arithmetic	387
13.26	Fused multiply-add with pair precision	388
13.27	Higher intermediate precision and the FMA	393
13.28	Fused multiply-add without pair precision	395
13.29	Fused multiply-add with multiple precision	401
13.30	Fused multiply-add, Boldo/Melquiond style	403
13.31	Error correction in fused multiply-add	406
13.32	Retrospective on pair-precision arithmetic	407
14	Power function	411
14.1	Why the power function is hard to compute	411
14.2	Special cases for the power function	412
14.3	Integer powers	414
14.4	Integer powers, revisited	420
14.5	Outline of the power-function algorithm	421
14.6	Finding a and p	423
14.7	Table searching	424
14.8	Computing $\log_n(g/a)$	426
14.9	Accuracy required for $\log_n(g/a)$	429
14.10	Exact products	430
14.11	Computing w , w_1 and w_2	433
14.12	Computing n^{w_2}	437
14.13	The choice of q	438
14.14	Testing the power function	438
14.15	Retrospective on the power function	440
15	Complex arithmetic primitives	441
15.1	Support macros and type definitions	442
15.2	Complex absolute value	443
15.3	Complex addition	445
15.4	Complex argument	445
15.5	Complex conjugate	446
15.6	Complex conjugation symmetry	446
15.7	Complex conversion	448
15.8	Complex copy	448
15.9	Complex division: C99 style	449
15.10	Complex division: Smith style	451
15.11	Complex division: Stewart style	452
15.12	Complex division: Priest style	453
15.13	Complex division: avoiding subtraction loss	455
15.14	Complex imaginary part	456

15.15	Complex multiplication	456
15.16	Complex multiplication: error analysis	458
15.17	Complex negation	459
15.18	Complex projection	460
15.19	Complex real part	460
15.20	Complex subtraction	461
15.21	Complex infinity test	462
15.22	Complex NaN test	462
15.23	Summary	463
16	Quadratic equations	465
16.1	Solving quadratic equations	465
16.2	Root sensitivity	471
16.3	Testing a quadratic-equation solver	472
16.4	Summary	474
17	Elementary functions in complex arithmetic	475
17.1	Research on complex elementary functions	475
17.2	Principal values	476
17.3	Branch cuts	476
17.4	Software problems with negative zeros	478
17.5	Complex elementary function tree	479
17.6	Series for complex functions	479
17.7	Complex square root	480
17.8	Complex cube root	485
17.9	Complex exponential	487
17.10	Complex exponential near zero	492
17.11	Complex logarithm	495
17.12	Complex logarithm near one	497
17.13	Complex power	500
17.14	Complex trigonometric functions	502
17.15	Complex inverse trigonometric functions	504
17.16	Complex hyperbolic functions	509
17.17	Complex inverse hyperbolic functions	514
17.18	Summary	520
18	The Greek functions: gamma, psi, and zeta	521
18.1	Gamma and log-gamma functions	521
18.1.1	Outline of the algorithm for <code>tgamma()</code>	525
18.1.1.1	Asymptotic expansions	528
18.1.1.2	Recurrence-relation accuracy	528
18.1.1.3	Sums of rational numbers	529
18.1.1.4	Avoiding catastrophic overflow	530
18.1.2	Gamma function accuracy	531
18.1.3	Computation of $\pi / \sin(\pi x)$	531
18.1.4	Why <code>lgamma(x)</code> is hard to compute accurately	531
18.1.5	Outline of the algorithm for <code>lgamma()</code>	534
18.1.6	Log-gamma function accuracy	536
18.2	The <code>psi()</code> and <code>psiln()</code> functions	536
18.2.1	Psi function poles and zeros	538
18.2.2	Recurrence relations for psi functions	539
18.2.3	Psi functions with negative arguments	541
18.2.4	Psi functions with argument multiples	541
18.2.5	Taylor-series expansions of psi functions	542
18.2.6	Asymptotic expansion of the psi function	542

18.2.7	Psi function on $[0, 1]$	543
18.2.8	Outline of the algorithm for $\text{psi}()$	543
18.2.9	Computing $\pi / \tan(\pi x)$	545
18.2.10	Outline of the algorithm for $\text{psi}\ln()$	546
18.3	Polygamma functions	547
18.3.1	Applications of polygamma functions	555
18.3.2	Computing the polygamma functions	556
18.3.3	Retrospective on the polygamma functions	558
18.4	Incomplete gamma functions	560
18.5	A Swiss diversion: Bernoulli and Euler	568
18.5.1	Bernoulli numbers revisited	574
18.6	An Italian excursion: Fibonacci numbers	575
18.7	A German gem: the Riemann zeta function	579
18.7.1	Computing the Riemann zeta function	583
18.7.2	Greek relatives of the Riemann zeta function	587
18.8	Further reading	590
18.9	Summary	591
19	Error and probability functions	593
19.1	Error functions	593
19.1.1	Properties of the error functions	593
19.1.2	Computing the error functions	595
19.2	Scaled complementary error function	598
19.3	Inverse error functions	600
19.3.1	Properties of the inverse error functions	601
19.3.2	Historical algorithms for the inverse error functions	603
19.3.3	Computing the inverse error functions	605
19.4	Normal distribution functions and inverses	610
19.5	Summary	617
20	Elliptic integral functions	619
20.1	The arithmetic-geometric mean	619
20.2	Elliptic integral functions of the first kind	624
20.3	Elliptic integral functions of the second kind	627
20.4	Elliptic integral functions of the third kind	630
20.5	Computing $K(m)$ and $K'(m)$	631
20.6	Computing $E(m)$ and $E'(m)$	637
20.7	Historical algorithms for elliptic integrals	643
20.8	Auxiliary functions for elliptic integrals	645
20.9	Computing the elliptic auxiliary functions	648
20.10	Historical elliptic functions	650
20.11	Elliptic functions in software	652
20.12	Applications of elliptic auxiliary functions	653
20.13	Elementary functions from elliptic auxiliary functions	654
20.14	Computing elementary functions via $R_C(x, y)$	655
20.15	Jacobian elliptic functions	657
20.15.1	Properties of Jacobian elliptic functions	659
20.15.2	Computing Jacobian elliptic functions	661
20.16	Inverses of Jacobian elliptic functions	664
20.17	The modulus and the nome	668
20.18	Jacobian theta functions	673
20.19	Logarithmic derivatives of the Jacobian theta functions	675
20.20	Neville theta functions	678
20.21	Jacobian Eta, Theta, and Zeta functions	679
20.22	Weierstrass elliptic functions	682

20.23	Weierstrass functions by duplication	689
20.24	Complete elliptic functions, revisited	690
20.25	Summary	691
21	Bessel functions	693
21.1	Cylindrical Bessel functions	694
21.2	Behavior of $J_n(x)$ and $Y_n(x)$	695
21.3	Properties of $J_n(z)$ and $Y_n(z)$	697
21.4	Experiments with recurrences for $J_0(x)$	705
21.5	Computing $J_0(x)$ and $J_1(x)$	707
21.6	Computing $J_n(x)$	710
21.7	Computing $Y_0(x)$ and $Y_1(x)$	713
21.8	Computing $Y_n(x)$	715
21.9	Improving Bessel code near zeros	716
21.10	Properties of $I_n(z)$ and $K_n(z)$	718
21.11	Computing $I_0(x)$ and $I_1(x)$	724
21.12	Computing $K_0(x)$ and $K_1(x)$	726
21.13	Computing $I_n(x)$ and $K_n(x)$	728
21.14	Properties of spherical Bessel functions	731
21.15	Computing $j_n(x)$ and $y_n(x)$	735
21.16	Improving $j_1(x)$ and $y_1(x)$	740
21.17	Modified spherical Bessel functions	743
21.17.1	Computing $i_0(x)$	744
21.17.2	Computing $is_0(x)$	746
21.17.3	Computing $i_1(x)$	747
21.17.4	Computing $is_1(x)$	749
21.17.5	Computing $i_n(x)$	750
21.17.6	Computing $is_n(x)$	753
21.17.7	Computing $k_n(x)$ and $ks_n(x)$	754
21.18	Software for Bessel-function sequences	755
21.19	Retrospective on Bessel functions	761
22	Testing the library	763
22.1	Testing <code>tgamma()</code> and <code>lgamma()</code>	765
22.2	Testing <code>psi()</code> and <code>psiln()</code>	768
22.3	Testing <code>erf()</code> and <code>erfc()</code>	768
22.4	Testing cylindrical Bessel functions	769
22.5	Testing exponent/significand manipulation	769
22.6	Testing inline assembly code	769
22.7	Testing with Maple	770
22.8	Testing floating-point arithmetic	773
22.9	The Berkeley Elementary Functions Test Suite	774
22.10	The AT&T floating-point test package	775
22.11	The Antwerp test suite	776
22.12	Summary	776
23	Pair-precision elementary functions	777
23.1	Pair-precision integer power	777
23.2	Pair-precision machine epsilon	779
23.3	Pair-precision exponential	780
23.4	Pair-precision logarithm	787
23.5	Pair-precision logarithm near one	793
23.6	Pair-precision exponential near zero	793
23.7	Pair-precision base- n exponentials	795
23.8	Pair-precision trigonometric functions	796

23.9	Pair-precision inverse trigonometric functions	801
23.10	Pair-precision hyperbolic functions	804
23.11	Pair-precision inverse hyperbolic functions	808
23.12	Summary	808
24	Accuracy of the Cody/Waite algorithms	811
25	Improving upon the Cody/Waite algorithms	823
25.1	The Bell Labs libraries	823
25.2	The Cephes library	823
25.3	The Sun libraries	824
25.4	Mathematical functions on EPIC	824
25.5	The GNU libraries	825
25.6	The French libraries	825
25.7	The NIST effort	826
25.8	Commercial mathematical libraries	826
25.9	Mathematical libraries for decimal arithmetic	826
25.10	Mathematical library research publications	826
25.11	Books on computing mathematical functions	827
25.12	Summary	828
26	Floating-point output	829
26.1	Output character string design issues	830
26.2	Exact output conversion	831
26.3	Hexadecimal floating-point output	832
26.3.1	Hexadecimal floating-point output requirements	832
26.3.2	Remarks on hexadecimal floating-point output	833
26.3.3	Hexadecimal floating-point output-conversion code	834
26.3.4	Conversion to uppercase	848
26.3.5	Determining rounding direction	848
26.4	Octal floating-point output	850
26.5	Binary floating-point output	851
26.6	Decimal floating-point output	851
26.6.1	The decimal-conversion program interface	853
26.6.2	Fast powers of ten	855
26.6.3	Preliminary scaling	856
26.6.4	Support for %g-style conversion	857
26.6.5	Buffer sizes	858
26.6.6	Special cases	858
26.6.7	Scaling and rounding adjustment	859
26.6.8	Digit generation	860
26.6.9	Completing decimal output conversion	861
26.6.10	Computing the minimum desirable precision	864
26.6.11	Coding fast powers of ten	864
26.7	Accuracy of output conversion	865
26.8	Output conversion to a general base	865
26.9	Output conversion of Infinity	866
26.10	Output conversion of NaN	866
26.11	Number-to-string conversion	867
26.12	The printf() family	867
26.12.1	Dangers of printf()	868
26.12.2	Variable argument lists	870
26.12.3	Implementing the printf() family	871
26.12.4	Output conversion specifiers	873
26.13	Summary	878

27 Floating-point input	879
27.1 Binary floating-point input	879
27.1.1 Sign input conversion	885
27.1.2 Prefix string matching	886
27.1.3 Infinity input conversion	887
27.1.4 NaN input conversion	887
27.1.5 Power input conversion	889
27.1.6 Floating-point suffix conversion	890
27.1.7 Integer suffix conversion	892
27.1.8 Input rounding adjustment	893
27.2 Octal floating-point input	894
27.3 Hexadecimal floating-point input	895
27.4 Decimal floating-point input	895
27.5 Based-number input	899
27.6 General floating-point input	900
27.7 The <code>scanf()</code> family	901
27.7.1 Implementing the <code>scanf()</code> family	902
27.7.2 Whitespace and ordinary characters	904
27.7.3 Input conversion specifiers	905
27.7.4 Retrospective on the <code>scanf()</code> family	909
27.8 Summary	910
A Ada interface	911
A.1 Building the Ada interface	911
A.2 Programming the Ada interface	912
A.3 Using the Ada interface	915
B C# interface	917
B.1 C# on the CLI virtual machine	917
B.2 Building the C# interface	918
B.3 Programming the C# interface	920
B.4 Using the C# interface	922
C C++ interface	923
C.1 Building the C++ interface	923
C.2 Programming the C++ interface	924
C.3 Using the C++ interface	925
D Decimal arithmetic	927
D.1 Why we need decimal floating-point arithmetic	927
D.2 Decimal floating-point arithmetic design issues	928
D.3 How decimal and binary arithmetic differ	931
D.4 Initialization of decimal floating-point storage	935
D.5 The <code><decimal.h></code> header file	936
D.6 Rounding in decimal arithmetic	936
D.7 Exact scaling in decimal arithmetic	937
E Errata in the Cody/Waite book	939
F Fortran interface	941
F.1 Building the Fortran interface	943
F.2 Programming the Fortran interface	944
F.3 Using the Fortran interface	945

H	Historical floating-point architectures	947
H.1	CDC family	949
H.2	Cray family	952
H.3	DEC PDP-10	953
H.4	DEC PDP-11 and VAX	956
H.5	General Electric 600 series	958
H.6	IBM family	959
H.6.1	IBM 7030 Stretch	959
H.6.2	IBM and Fortran	961
H.6.3	IBM System/360	963
H.7	Lawrence Livermore S-1 Mark IIA	965
H.8	Unusual floating-point systems	966
H.9	Historical retrospective	967
I	Integer arithmetic	969
I.1	Memory addressing and integers	971
I.2	Representations of signed integers	971
I.2.1	Sign-magnitude representation	971
I.2.2	One's-complement representation	972
I.2.3	Two's-complement representation	972
I.2.4	Excess- <i>n</i> representation	974
I.2.5	Ranges of integers	974
I.3	Parity testing	975
I.4	Sign testing	975
I.5	Arithmetic exceptions	975
I.6	Notations for binary numbers	977
I.7	Summary	978
J	Java interface	979
J.1	Building the Java interface	979
J.2	Programming the Java MathCW class	980
J.3	Programming the Java C interface	982
J.4	Using the Java interface	985
L	Letter notation	987
P	Pascal interface	989
P.1	Building the Pascal interface	989
P.2	Programming the Pascal MathCW module	990
P.3	Using the Pascal module interface	993
P.4	Pascal and numeric programming	994
	Bibliography	995
	Author/editor index	1039
	Function and macro index	1049
	Subject index	1065
	Colophon	1115

The Mathematical-Function Computation Handbook
Programming Using the MathCW Portable Software
Library

Beebe, N.H.F.

2017, XXXVI, 1115 p., Hardcover

ISBN: 978-3-319-64109-6