

Contents

<i>Preface</i>	<i>page ix</i>
Part I Numerical Software	1
1 Why <i>numerical</i> software?	3
1.1 Efficient kernels	4
1.2 Rapid change	5
1.3 Large-scale problems	6
2 Scientific computation and numerical analysis	8
2.1 The trouble with real numbers	8
2.2 Fixed-point arithmetic	18
2.3 Algorithm stability vs. problem stability	19
2.4 Numerical accuracy and reliability	23
3 Priorities	30
3.1 Correctness	30
3.2 Numerical stability	32
3.3 Accurate discretization	32
3.4 Flexibility	33
3.5 Efficiency: time and memory	35
4 Famous disasters	36
4.1 Patriot missiles	36
4.2 Ariane 5	37
4.3 Sleipner A oil rig collapse	38
5 Exercises	39
Part II Developing Software	43
6 Basics of computer organization	45
6.1 Under the hood: what a CPU does	45
6.2 Calling routines: stacks and registers	47
6.3 Allocating variables	51
6.4 Compilers, linkers, and loaders	53

vi	<i>Contents</i>	
7	Software design	57
7.1	Software engineering	57
7.2	Software life-cycle	57
7.3	Programming in the large	59
7.4	Programming in the small	61
7.5	Programming in the middle	67
7.6	Interface design	70
7.7	Top-down and bottom-up development	75
7.8	Don't hard-wire it unnecessarily!	77
7.9	Comments	78
7.10	Documentation	80
7.11	Cross-language development	82
7.12	Modularity and all that	87
8	Data structures	90
8.1	Package your data!	90
8.2	Avoid global variables!	91
8.3	Multidimensional arrays	92
8.4	Functional representation vs. data structures	96
8.5	Functions and the "environment problem"	97
8.6	Some comments on object-oriented scientific software	106
9	Design for testing and debugging	118
9.1	Incremental testing	118
9.2	Localizing bugs	120
9.3	The mighty "print" statement	120
9.4	Get the computer to help	122
9.5	Using debuggers	129
9.6	Debugging functional representations	130
9.7	Error and exception handling	132
9.8	Compare and contrast	135
9.9	Tracking bugs	136
9.10	Stress testing and performance testing	137
9.11	Random test data	141
10	Exercises	143
	Part III Efficiency in Time, Efficiency in Memory	147
11	Be algorithm aware	149
11.1	Numerical algorithms	149
11.2	Discrete algorithms	151
11.3	Numerical algorithm design techniques	153
12	Computer architecture and efficiency	156
12.1	Caches and memory hierarchies	156

Contents

vii

12.2	A tour of the Pentium 4 TM architecture	158
12.3	Virtual memory and paging	164
12.4	Thrashing	164
12.5	Designing for memory hierarchies	165
12.6	Dynamic data structures and memory hierarchies	168
12.7	Pipelining and loop unrolling	168
12.8	Basic Linear Algebra Software (BLAS)	170
12.9	LAPACK	178
12.10	Cache-oblivious algorithms and data structures	184
12.11	Indexing vs. pointers for dynamic data structures	185
13	Global vs. local optimization	187
13.1	Picking algorithms vs. keyhole optimization	187
13.2	What optimizing compilers do	188
13.3	Helping the compiler along	191
13.4	Practicalities and asymptotic complexity	192
14	Grabbing memory when you need it	195
14.1	Dynamic memory allocation	195
14.2	Giving it back	197
14.3	Garbage collection	198
14.4	Life with garbage collection	199
14.5	Conservative garbage collection	202
14.6	Doing it yourself	203
14.7	Memory tips	205
15	Memory bugs and leaks	208
15.1	Beware: unallocated memory!	208
15.2	Beware: overwriting memory!	208
15.3	Beware: dangling pointers!	210
15.4	Beware: memory leaks!	214
15.5	Debugging tools	215
Part IV	Tools	217
16	Sources of scientific software	219
16.1	Netlib	220
16.2	BLAS	220
16.3	LAPACK	221
16.4	GAMS	221
16.5	Other sources	221
17	Unix tools	223
17.1	Automated builds: make	223
17.2	Revision control: RCS, CVS, Subversion and Bitkeeper	226

viii	<i>Contents</i>	
17.3	Profiling: prof and gprof	228
17.4	Text manipulation: grep, sed, awk, etc.	230
17.5	Other tools	232
17.6	What about Microsoft Windows?	233
Part V	Design Examples	237
18	Cubic spline function library	239
18.1	Creation and destruction	242
18.2	Output	244
18.3	Evaluation	244
18.4	Spline construction	247
18.5	Periodic splines	257
18.6	Performance testing	260
19	Multigrid algorithms	262
19.1	Discretizing partial differential equations	262
19.2	Outline of multigrid methods	264
19.3	Implementation of framework	265
19.4	Common choices for the framework	272
19.5	A first test	273
19.6	The operator interface and its uses	276
19.7	Dealing with sparse matrices	279
19.8	A second test	282
Appendix A	Review of vectors and matrices	287
A.1	Identities and inverses	288
A.2	Norms and errors	289
A.3	Errors in solving linear systems	291
Appendix B	Trademarks	292
	<i>References</i>	293
	<i>Index</i>	299