
Contents

Preface.....	xix
Author.....	xxv
1. Introduction to Performance Tuning.....	1
1.1 Why Should We Bother?	1
1.2 When to Performance-Tune and When Not to Bother.....	3
1.3 The Iterative Performance-Tuning Cycle	5
1.3.1 Pareto's Principle and the Law of Diminishing Returns	6
1.3.2 When to Stop Tuning.....	7
1.3.3 Periodic Performance Maintenance	9
1.4 What to Tune.....	9
1.5 Performance Tuning Pitfalls.....	10
1.5.1 When to Tune	10
1.5.2 Performance Goals.....	10
1.5.3 Profiling.....	11
1.5.4 Optimization	11
1.6 Performance Tuning Tradeoffs.....	13
1.7 Vertical versus Horizontal Scaling.....	16
1.8 Perceived versus Actual Performance.....	18
1.8.1 Presenting Continuous Feedback for Ongoing Tasks	19
1.8.2 Placing the User in Control	20
1.8.3 Enabling User Interaction during Background Processing.....	20
1.8.4 Streaming Data as It Becomes Available	21
1.8.5 Streamlining the Application.....	21
1.8.6 Reducing the Run-Time Variability.....	22
1.8.7 Performance and Real Time	22
2. Profiling MATLAB® Performance	25
2.1 The MATLAB Profiler	26
2.1.1 The Detailed Profiling Report.....	28
2.1.2 A Sample Profiling Session	32
2.1.3 Programmatic Access to Profiling Data	37
2.1.4 Function-Call History Timeline	39
2.1.5 CPU versus Wall-Clock Profiling.....	42
2.1.6 Profiling Techniques.....	42
2.1.7 Profiling Limitations	48
2.1.8 Profiling and MATLAB's JIT	49
2.2 <i>tic, toc</i> and Relatives	51
2.2.1 The Built-In <i>tic, toc</i> Functions	51
2.2.2 Comparison between the Profiler and <i>tic, toc</i>	53
2.2.3 Related Tools.....	54
2.3 Timed Log Files and Printouts.....	56
2.4 Non-MATLAB Tools.....	57

3. Standard Performance-Tuning Techniques	59
3.1 Loop Optimization	60
3.1.1 Move Loop-Invariant Code Out of the Loop	60
3.1.2 Minimize Function Call Overheads	64
3.1.3 Employ Early Bail-Outs.....	65
3.1.4 Simplify Loop Contents	67
3.1.5 Unroll Simple Loops.....	69
3.1.6 Optimize Nested Loops.....	70
3.1.7 Switch the Order of Nested Loops	72
3.1.8 Minimize Dereferencing.....	74
3.1.9 Postpone I/O and Graphics until the Loop Ends.....	75
3.1.10 Merge or Split Loops	75
3.1.11 Loop Over the Shorter Dimension	76
3.1.12 Run Loops Backwards	77
3.1.13 Partially Optimize a Loop.....	77
3.1.14 Use the Loop Index Rather than Counters.....	78
3.1.15 MATLAB's JIT.....	78
3.2 Data Caching	81
3.2.1 Read-Only Caches	82
3.2.2 Common Subexpression Elimination	83
3.2.3 Persistent Caches.....	83
3.2.4 Writable Caches.....	85
3.2.5 A Real-Life Example: Writable Cache	88
3.2.6 Optimizing Cache Fetch Time.....	90
3.3 Smart Checks Bypass	92
3.4 Exception Handling.....	95
3.5 Improving Externally Connected Systems.....	97
3.5.1 Database	97
3.5.2 File System and Network.....	105
3.5.3 Computer Hardware	107
3.6 Processing Smaller Data Subsets	109
3.6.1 Reading from a Database.....	109
3.6.2 Reading from a Data File	110
3.6.3 Processing Data.....	111
3.7 Interrupting Long-Running Tasks	113
3.8 Latency versus Throughput	115
3.8.1 Lazy Evaluation.....	115
3.8.2 Prefetching.....	118
3.9 Data Analysis.....	119
3.9.1 Preprocessing the Data	120
3.9.2 Controlling the Target Accuracy	122
3.9.3 Reducing Problem Complexity	123
3.10 Other Techniques	126
3.10.1 Coding	126
3.10.2 Data.....	132
3.10.3 General	133
4. MATLAB®-Specific Techniques
4.1 Effects of Using Different Data Types.....

4.1.1	Numeric versus Nonnumeric Data Types.....	135
4.1.2	Nondouble and Multidimensional Arrays.....	136
4.1.3	Sparse Data	137
4.1.4	Modifying Data Type in Run Time.....	140
4.1.5	Concatenating Cell Arrays	141
4.1.6	Datasets, Tables, and Categorical Arrays	142
4.1.7	Additional Aspects	142
4.2	Characters and Strings.....	143
4.2.1	MATLAB's Character/Number Duality.....	144
4.2.2	Search and Replace.....	144
4.2.3	Converting Numbers to Strings (and Back).....	149
4.2.4	String Comparison.....	150
4.2.5	Additional Aspects.....	152
4.3	Using Internal Helper Functions.....	153
4.3.1	A Sample Debugging Session	154
4.4	Date and Time Functions.....	157
4.5	Numeric Processing.....	162
4.5.1	Using <code>inf</code> and <code>NaN</code>	162
4.5.2	Matrix Operations.....	163
4.5.3	Real versus Complex Math.....	168
4.5.4	Gradient.....	169
4.5.5	Optimization	170
4.5.6	Fast Fourier Transform.....	175
4.5.7	Updating the Math Libraries.....	177
4.5.8	Random Numbers	180
4.6	Functional Programming	181
4.6.1	Invoking Functions.....	181
4.6.2	<i>onCleanup</i>	188
4.6.3	Conditional Constructs.....	189
4.6.4	Smaller Functions and M-files	190
4.6.5	Effective Use of the MATLAB Path.....	191
4.6.6	Overloaded Built-In MATLAB Functions	191
4.7	Object-Oriented MATLAB.....	193
4.7.1	Object Creation.....	193
4.7.2	Accessing Properties	195
4.7.3	Invoking Methods.....	199
4.7.4	Using System Objects	201
4.8	MATLAB Start-Up	202
4.8.1	The MATLAB Startup Accelerator	202
4.8.2	Starting MATLAB in Batch Mode	206
4.8.3	Slow MATLAB Start-Up.....	206
4.8.4	Profiling MATLAB Start-Up.....	208
4.8.5	Java Start-Up	209
4.9	Additional Techniques	210
4.9.1	Reduce the Number of Workspace Variables.....	210
4.9.2	Loop Over the Smaller Data Set	211
4.9.3	Referencing Dynamic Struct Fields and Object Properties.....	212
4.9.4	Use Warning with a Specific Message ID	213
4.9.5	Prefer <i>num2cell</i> Rather than <i>mat2cell</i>	213

4.9.6	Avoid Using <i>containers.Map</i>	213
4.9.7	Use the Latest MATLAB Release and Patches	215
4.9.8	Use <i>is*</i> Functions Where Available	216
4.9.9	Specify the Item Type When Using <i>ishghandle</i> or <i>exist</i>	216
4.9.10	Use Problem-Specific Tools	217
4.9.11	Symbolic Arithmetic	217
4.9.12	Simulink	217
4.9.13	Mac OS	219
4.9.14	Additional Ideas	221
5	Implicit Parallelization (Vectorization and Indexing)	223
5.1	Introduction to MATLAB® Vectorization	223
5.1.1	So What Exactly Is MATLAB Vectorization?	223
5.1.2	Indexing Techniques	226
5.1.3	Logical Indexing	229
5.2	Built-In Vectorization Functions	233
5.2.1	Functions for Common Indexing Usage Patterns	233
5.2.2	Functions That Create Arrays	234
5.2.3	Functions That Accept Vectorized Data	234
5.2.4	Functions That Apply Another Function in a Vectorized Manner	238
5.2.5	Set-Based Functions	245
5.3	Simple Vectorization Examples	247
5.3.1	Trivial Transformations	247
5.3.2	Partial Data Summation	248
5.3.3	Thresholding	249
5.3.4	Cumulative Sum	250
5.3.5	Data Binning	251
5.3.6	Using <i>meshgrid</i> and <i>bsxfun</i>	251
5.3.7	A <i>meshgrid</i> Variant	252
5.3.8	Euclidean Distances	253
5.3.9	Range Search	254
5.3.10	Matrix Computations	255
5.4	Repetitive Data	256
5.4.1	A Simple Example	258
5.4.2	Using <i>repmat</i> Replacements	259
5.4.3	Repetitions of Internal Elements	260
5.5	Multidimensional Data	261
5.6	Real-Life Example: Synthetic Aperture Radar Matched Filter	264
5.6.1	Naïve Approach	264
5.6.2	Using Vectorization	266
5.7	Effective Use of MATLAB Vectorization	268
5.7.1	Vectorization Is Not Always Faster	268
5.7.2	Applying Smart Indexing	269
5.7.3	Breaking a Problem into Simpler Vectorizable Subproblems	270
5.7.4	Using Vectorization as Replacement for Iterative Data Updates	271
5.7.5	Minimizing Temporary Data Allocations	271
5.7.6	Preprocessing Inputs, Rather than Postprocessing the Output	272
5.7.7	Interdependent Loop Iterations	272
5.7.8	Reducing Loop Complexity	274

5.7.9	Reducing Processing Complexity.....	276
5.7.10	Nested Loops.....	276
5.7.11	Analyzing Loop Pattern to Extract a Vectorization Rule.....	277
5.7.12	Vectorizing Structure Elements.....	278
5.7.13	Limitations of Internal Parallelization.....	279
5.7.14	Using MATLAB's Character/Number Duality	280
5.7.15	Acklam's Vectorization Guide and Toolbox.....	281
5.7.16	Using Linear Algebra to Avoid Looping Over Matrix Indexes	281
5.7.17	Intersection of Curves: Reader Exercise	282
6	Explicit Parallelization Using MathWorks Toolboxes	285
6.1	The Parallel Computing Toolbox — CPUs	286
6.1.1	Using <i>parfor</i> -Loops	287
6.1.2	Using <i>spmd</i>	292
6.1.3	Distributed and Codistributed Arrays	296
6.1.4	Interactive Parallel Development with <i>pmode</i>	300
6.1.5	Profiling Parallel Blocks.....	302
6.1.6	Running Example: Using <i>parfor</i> Loops	305
6.1.7	Running Example: Using <i>spmd</i>	306
6.2	The Parallel Computing Toolbox — GPUs.....	308
6.2.1	Introduction to General-Purpose GPU Computing.....	308
6.2.2	Parallel Computing with GPU Arrays.....	310
6.2.3	Running Example: Using GPU Arrays	316
6.2.4	Running Example: Using Multiple GPUs with <i>spmd</i> Construct.....	317
6.2.5	Executing CUDA Kernels from MATLAB	319
6.2.6	Running Example: Using CUDA Kernels	322
6.2.7	Programming GPU Using MATLAB MEX	324
6.2.8	Accessing GPUs from within Parallel Blocks	327
6.3	The MATLAB Distributed Computing Server.....	328
6.3.1	Using MDCS.....	329
6.3.2	Parallel Jobs Overview	331
6.3.3	Setting Up a Scheduler Interface	333
6.3.4	Programming Independent Jobs	336
6.3.5	Programming Communicating Jobs.....	337
6.3.6	Using Batch Processing on a Cluster.....	339
6.3.7	Running Example: Using Communicating Jobs on a Cluster	340
6.4	Techniques for Effective Parallelization in MATLAB	342
6.4.1	General Performance Tips	342
6.4.2	Performance Tips for Parallel CPU Programming	345
6.4.3	Performance Tips for Parallel GPU Programming	348
7	Explicit Parallelization by Other Means.....	353
7.1	GPU Acceleration Using Jacket.....	353
7.1.1	Key Ideas of Jacket Design.....	353
7.1.2	Jacket Interface to MATLAB.....	354
7.1.3	Using Parallel <i>gfor</i> Loops	356
7.1.4	Compiling M-Code to a CUDA Kernel with <i>gcompile</i>	357
7.1.5	Multi-GPU Support	358
7.1.6	Running Example: Using Parallel <i>gfor</i> -Loop	360

7.1.7	Running Example: Using <i>gcompile</i>	361
7.1.8	Running Example: Using <i>spmd</i> and Multi-GPU Support	362
7.2	Alternative/Related Technologies	364
7.2.1	Using GPUmat.....	364
7.2.2	Multicore Library for Parallel Processing on Multiple Cores.....	368
7.2.3	Using ArrayFire Library via MEX Interface	370
7.2.4	Additional Alternatives	373
7.3	Multithreading	374
7.3.1	Using POSIX Threads.....	374
7.3.2	Using OpenMP.....	377
7.3.3	Using Java Threads.....	379
7.3.4	Using .Net Threads.....	382
7.3.5	Using MATLAB Timers	384
7.4	Spawning External Processes.....	385
8	Using Compiled Code.....	389
8.1	Using MEX Code.....	389
8.1.1	Introduction.....	389
8.1.2	Our First MEX Function	390
8.1.3	MEX Function Inputs and Outputs.....	394
8.1.4	Accessing MATLAB Data	395
8.1.5	A Usage Example	401
8.1.6	Memory Management.....	404
8.1.7	Additional Aspects	406
8.2	Using the MATLAB Coder Toolbox	413
8.2.1	Code Adaptation	414
8.2.2	A Simple Example: Euclidean Distances Algorithm	416
8.2.3	A More Realistic Example: Dijkstra's Shortest-Path Algorithm.....	420
8.2.4	Configuring the Coder for Maximal Performance	427
8.3	Porting MATLAB Algorithms to FPGA.....	433
8.3.1	Algorithm Adaptation.....	434
8.3.2	HDL Workflow	435
8.3.3	Run-Time Measurements.....	444
8.4	Deployed (Compiled) MATLAB Programs	445
8.5	Using External Libraries	450
8.5.1	Introduction.....	450
8.5.2	Java	451
8.5.3	NAG: Numerical Algorithms Group.....	452
8.5.4	MATLAB Toolbox	455
8.5.5	MCT: Multi-Precision Computing Toolbox	456
8.5.6	Additional Libraries	457
9	Memory-Related Techniques	459
9.1	Why Memory Affects Performance.....	459
9.2	Profiling Memory Usage	461
9.2.1	Workspace Browser	461
9.2.2	<i>whos</i> Function.....	462
9.2.3	<i>memory</i> Function	463
9.2.4	<i>feature memstats</i> and <i>feature dumpmem</i>	464

9.2.5	<i>feature mtic/mtoc</i>	466
9.2.6	Profiler's Memory-Monitoring Feature.....	467
9.2.7	Profiling Java Memory	469
9.2.8	Using Third-Party Tools	470
9.2.9	Using the Operating System's Tools.....	472
9.2.10	<i>format debug</i>	473
9.3	MATLAB's Memory Storage and Looping Order	475
9.3.1	Memory Storage of MATLAB Array Data.....	475
9.3.2	Loop Down Columns Rather than Rows	477
9.3.3	Effect of Subindexing	480
9.4	Array Memory Allocation	481
9.4.1	Dynamic Array Growth.....	481
9.4.2	Effects of Incremental JIT Improvements.....	482
9.4.3	Preallocate Large Data Arrays	483
9.4.4	Preallocation Location within the Code.....	488
9.4.5	Preallocating Nondouble Data.....	488
9.4.6	Alternatives for Enlarging Arrays.....	491
9.5	Minimizing Memory Allocations.....	493
9.5.1	MATLAB's Copy-on-Write Mechanism.....	493
9.5.2	In-Place Data Manipulations.....	496
9.5.3	Reusing Variables (with Utmost Care).....	502
9.5.4	Clearing Unused Workspace Variables	505
9.5.5	Global and Persistent Variables	507
9.5.6	Scoping Rules and Nested Functions	509
9.5.7	Passing Handle References (Not Data) to Functions.....	511
9.5.8	Reducing Data Precision/Type.....	512
9.5.9	Devectoring Huge Data Operations	513
9.5.10	Assign Anonymous Functions in Dedicated Wrapper Functions.....	514
9.5.11	Represent Objects by Simpler Data Types.....	516
9.6	Memory Packing	518
9.7	Additional Recommendations	519
9.7.1	MATLAB Variables.....	519
9.7.2	Java Objects.....	522
10	Graphics and GUI	525
10.1	Initial Graphs Generation	526
10.1.1	Reduce the Number of Plotting Elements.....	526
10.1.2	Use Simple or No Plot Markers.....	530
10.1.3	Use Vectorized Data for Plotting.....	532
10.1.4	Use Static Axes Properties	533
10.1.5	Only Use <i>drawnow</i> when You Are Finished Plotting	534
10.1.6	Use Low-Level Rather than High-Level Graphic Functions	536
10.1.7	Generate Plots while the Figure Is Hidden.....	537
10.1.8	Apply Data Reduction to Plotted Data to Fit the Display Area	538
10.1.9	Reuse Plot Axes.....	540
10.1.10	Avoid Using the <i>axes</i> Function.....	540
10.1.11	Use the Painters Figure Renderer with Fast Axes DrawMode.....	542
10.1.12	Images.....	543
10.1.13	Patches and Volume Surfaces.....	545

10.1.14	Colorbars	546
10.1.15	Legends	546
10.1.16	Reopen Presaved Figure and/or Plot Axes	547
10.1.17	Set Axes SortMethod to Childorder and Reduce Transparency.....	548
10.2	Updating Graphs and Images in Real Time.....	549
10.2.1	Axes Update.....	549
10.2.2	Plot and Image Update.....	550
10.2.3	Legends and Colorbars	552
10.2.4	Accessing Object Properties.....	553
10.2.5	Listeners and Callbacks	554
10.2.6	Trading Accuracy for Speed.....	555
10.2.7	Avoid Update to the Same Value	557
10.2.8	Cache Graphic Handles	557
10.2.9	Avoid Interlacing Property <i>get</i> and <i>set</i>	558
10.2.10	Use <i>hgtransform</i> to Transform Graphic Objects	559
10.3	Figure Window Performance Aspects.....	559
10.3.1	Use Hardware-Accelerated OpenGL Renderer and Functionality	560
10.3.2	Set a Nondefault WVisual/XVisual Property Value	561
10.3.3	Disable BackingStore.....	562
10.3.4	Disable DoubleBuffer	562
10.3.5	Set a Manual DitherMapMode on Old Platforms	563
10.3.6	Reuse Figure Windows	563
10.3.7	Sharing Data between GUI Callback Functions	564
10.3.8	Disable Anti-Aliasing	564
10.3.9	Use Smaller and Fewer Figure Windows.....	564
10.4	GUI Preparation and Responsiveness.....	565
10.4.1	Creating the Initial GUI.....	565
10.4.2	Presenting User Feedback	575
10.4.3	Performing Asynchronous Actions	584
10.5	Avoiding Common Pitfalls	591
10.5.1	Minimize Intentional Pauses	591
10.5.2	Delete Unused Graphic Objects.....	592
11	I/O Techniques	593
11.1	Reducing the Amount of I/O	593
11.2	Avoiding Repeated File Access	595
11.3	Reading and Writing Files	597
11.3.1	Text versus Binary Format.....	597
11.3.2	Text File Pre-Processing	598
11.3.3	Memory-Mapped Files	598
11.3.4	Reading Files Efficiently	600
11.3.5	Writing Files Efficiently	604
11.4	Data Compression and the <i>save</i> Function	607
11.5	Excel Files (and Microsoft Office Files in General)	614
11.6	Image Files	620
11.7	Using Java and C I/O.....	622
11.8	Searching, Parsing, and Comparing Files	625
11.8.1	Searching for Files	625
11.8.2	Parsing and Scanning Files	626

- 11.9 Additional Aspects 627
 - 11.9.1 Using p-Code..... 627
 - 11.9.2 Network I/O 628
 - 11.9.3 Miscellaneous..... 630
- Appendix A: Additional Resources 631**
- Appendix B: Performance Tuning Checklist 643**
- References and Notes..... 645**
- Index 723**