

**MATLAB Performance Tuning and Acceleration**

**Course Number:** MTLB-112
**Duration:** 1 day

**Overview**

This intermediate-level MATLAB Performance Tuning and Acceleration training course teaches attendees how to improve the run-time performance (speed) of their MATLAB programs. Students learn how to leverage tools for profiling and measuring performance. In addition, students learn a wide variety of speedup techniques, including loop optimization, data caching and chunking, I/O optimization, mathematical/physical identities, storage type modifications, compiled (binary) code, vectorization, parallelization, graphics, memory-related techniques, and more.

**Note:** This course can be condensed to a half-day with fewer examples and less hands-on practice. We strongly recommend the full-day version if possible.

**Prerequisites**

Attendees should have taken Accelebrate's [From MATLAB Scripts to Complete Programs course](file:////training/matlab-scripts-programs) or have equivalent knowledge. Students should already be comfortable using the MATLAB environment and have at least basic MATLAB programming experience.

**Materials**

All MATLAB training students will receive comprehensive courseware.

**Software Needed on Each Student PC**

* Any Windows, Linux, or macOS operating system
* A recent version of MATLAB

**Objectives**

* Understand tradeoffs in performance and cost-effectiveness of MATLAB code
* Learn how to measure and profile MATLAB’s run-time performance
* Discover top performance hotspots/bottlenecks in run-time code
* Acquire soft-skills such as knowing when to optimize and when not to bother
* Discover multiple possible ways of improving MATLAB run-time speed
* Understand memory’s effects on performance, and how to use this information
* Write MATLAB code that is highly performant and responsive
* Learn how to improve perceived speedup when actual speedup is not possible

**Outline**

* Introduction
* Profiling MATLAB Performance
	+ When to profile and when not to bother
	+ When should we stop optimizing the code?
	+ Profiling techniques
	+ Real-time profiling limitations
	+ MATLAB’s JIT and its effect on profiling
	+ Trade-offs: performance, maintainability, robustness, development cost, etc.
	+ Vertical vs. horizontal scalability
* Standard Programming Techniques
	+ Loop optimizations
	+ Caching data
	+ Smart checks bypass
	+ Exception handling and performance
	+ Sizing data sets
	+ Inlining code
	+ Externally-connected systems
	+ Perceived vs. actual performance
	+ Using mathematical identities
* MATLAB-Specific Techniques
	+ Using different storage types
	+ Object-orient MATLAB and performance
	+ Using internal helper functions
	+ Strings and dates/times
	+ MATLAB’s Startup Accelerator
* Using Binary Code
	+ Mex
	+ MATLAB Compiler vs. MATLAB Coder
	+ 3rd-party libraries
* I/O Speedup Techniques
	+ XLS/CSV read/write
	+ Binary vs. text format
	+ Reducing disk access
	+ Buffered, consolidated and chunked I/O
* Vectorization and Parallelization Techniques
	+ Vertical vs. horizontal scaling
	+ Parallelization mechanisms in MATLAB
	+ Vectorization
	+ Explicit parallelization
	+ Amdahl’s Law
	+ Using the GPU
* Graphics and GUI Techniques
	+ Initial graphs creation
	+ Updating graphs in real-time
	+ GUI preparation
	+ GUI responsiveness
	+ Feedback for long-duration tasks
	+ Dynamic updates/refresh
	+ Asynchronous updates/refresh
	+ Avoiding common pitfalls
* Memory-Related Techniques
	+ Why memory affects performance
	+ Profiling memory usage in MATLAB
	+ MATLAB’s memory storage
	+ Optimizing loop ordering
	+ Pre-allocation of data
	+ Minimizing run-time memory allocations
	+ In-place data manipulations
	+ Using global and persistent variables
* Conclusion