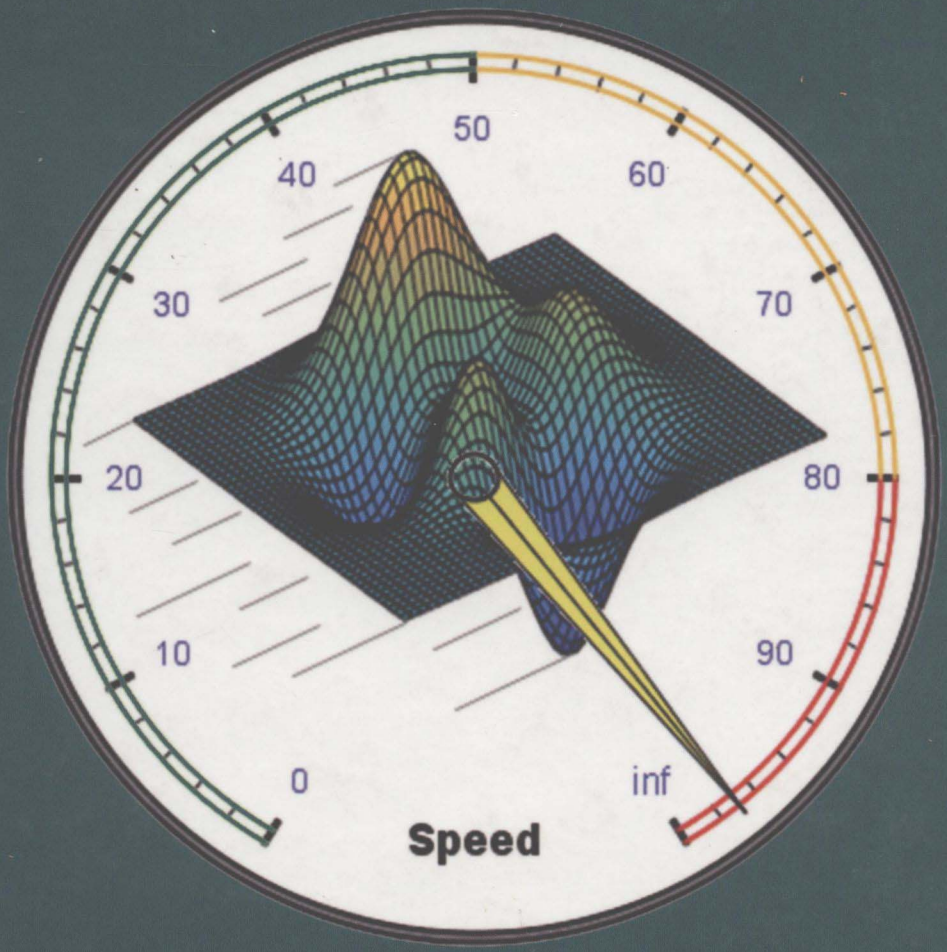
Accelerating MATLAB® [redacted]rformance

1001 tips to speed up MATLAB programs



Yair Altman

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Accelerating MATLAB® Performance

1001 tips to speed up MATLAB programs

Yair Altman

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed on acid-free paper
Version Date: 20141031

International Standard Book Number-13: 978-1-4822-1129-0 (Hardback)

Accelerating MATLAB® Performance

1001 tips to speed up MATLAB programs

To Tovi, Gali, Liat, and Lavi

Preface

The MATLAB® programming environment is often perceived as a platform suitable for prototyping and modeling but not for actual real-life applications. One of the reasons that I constantly hear when consulting with clients is that *"MATLAB is slow"*.

This book aims to help reduce this perception and shows that MATLAB programs can in fact be made to run extremely fast, in a wide variety of different ways.

MathWorks, who develop MATLAB, invests a significant amount of R&D effort in constantly improving MATLAB's performance and advocating best practices for improved performance.[1] Postings for performance-related R&D jobs are periodically posted[2] and the engine's performance improves with almost every semi-annual MATLAB release. In fact, the same MATLAB programs that might have been slow 10 or more years ago may now be blazingly fast when run using the latest MATLAB release, on the very same platform.

Using programming techniques presented in this book, MATLAB applications can be made even faster, fast enough for most uses. This enables significant reduction of the development time and cost, since we can use MATLAB from end to end, from prototyping to deployment, without having to maintain a mirror code–base using a different programming language and environment.

So, the perception of MATLAB as a slow environment may at least partly be due to a combination of factors, ranging from negative experience from past releases, to application code that does not follow good programming practices.

Some people say that when performance is really important, we should use a different programming language such as C/C++. While using C/C++ can certainly improve performance if done correctly, it is not a general panacea. MATLAB provides many benefits that could be very important during both development (e.g., rapid application development, short development cycles, ease of use, lenient environment, gentle learning curve) and run time (e.g., built-in vectorization, simple parallelization, automated memory management, and JIT optimizations). This is the reason we use High-Level Languages in the first place, trading some performance for functionality, development time, and so on. Otherwise, we would program in Assembly, use FPGAs, or even develop custom ASIC chips for top performance....

Using MATLAB does not however mean that we should abandon performance altogether. This book shows that using some very easy-to-follow techniques we can significantly improve MATLAB code speed without sacrificing MATLAB's benefits.

Some authors who write about performance like to demonstrate ideas with colorful graphs that show the performance behavior as a function of one or more parameters. Such comparisons are often academic in nature. This book tries to take a more practical approach with the presented recommendations. The ideas are explained verbally and short code snippets are often included to illustrate the point, usually without a rigorous comparison of all the parameter variants. This is not a PhD thesis, but rather a practical hands-on book intended for day-to-day use by engineers.

On one hand, enough information is provided to enable engineers to immediately apply the suggestions to their MATLAB programs. On the other hand, many references are also provided to enable readers who wish to expand the treatment of a particular topic to easily do so. There is always a delicate line between providing too much and too little information in the main text, so I hope my choices were adequate.

Performance is a term that can refer to many things, from functional (*"Does it perform well enough?"*) to speed (*"Does it perform fast enough?"*). In this book we are interested in only the latter aspect: speed. Other aspects of performance (accuracy, stability, robustness, etc.) are not less important, but are outside the scope of this text.

If this book may seem verbose at times, this is because I have tried to explain the reasons behind the recommendations, in the hope that users will gain insight. After all, this book cannot cover every possible aspect in all possible situations; gaining insight will enable readers to search for other ways to tune their specific program. Naturally, not all the numerous individual speedup suggestions can be remembered. But in my experience, once we understand and internalize the reasoning, we naturally "rediscover" these techniques whenever we come across a situation that merits them.

This book contains a wide variety of suggestions. Some may not be relevant for a specific application, or we may decide not to use them for some reason (e.g., due to their extra development and/or maintenance cost). Do not despair—plenty of other suggestions are available that could be helpful. There are many ways to achieve our target performance goals, so even if one technique fails, there are alternatives that we could try. In fact, there are so many different ways to achieve these goals that we can take a pick based on aesthetic preferences and subjective experience: Some people use vectorization, others like parallelization, some others prefer to invest in smarter algorithms, others trade memory for performance, still others display a GUI that gives the impression of being fast.

All of these routes and more are valid alternatives for making a program answer user expectations of speed and responsiveness. Moreover, it is expected that readers will become more proficient in efficient programming techniques, such that their programs will run faster in the first place, even before any tuning is actually done.

The book is meant as a generic reference for MATLAB performance tuning. As such, it does not include detailed discussion of domain-specific topics such as numerical analysis, optimization, statistics, algorithms, or image processing. These topics are well worth discussing for performance aspects, but they too are outside the scope of this book. Some discussion is included, but is not intended to be comprehensive nor detailed. Interested readers are referred to dedicated works on these specific topics.

Book Organization

This book is organized in chapters grouped by related functionality/usage. It is not necessary to read the book in order: the chapters and sections are mostly independent and stand alone. You can safely skip parts that you find difficult or uninteresting.

We begin with a theoretical description of performance tuning in Chapter 1. The discussion includes typical pitfalls, tradeoffs, and considerations that need to be kept in mind before and during any tuning process. Chapter 1 is not meant to be a comprehensive discussion of the theory of performance tuning; there are other books fully devoted to this subject. In contrast, I attempted to describe the essence of the major practical issues as I see them. It should be noted that this is not an exact science, and my subjective opinions on tradeoff considerations may well be disputed by others. Still, I hope that by reading this chapter, readers will at least be exposed to the underlying questions and considerations that relate to performance tuning, even if they disagree with my analysis or recommendations.

As long as you keep the underlying questions in mind when you tune an application, you should be okay.

Chapter 2 provides an overview of tools that are available in MATLAB in order to diagnose an application to determine the locations of, and reasons for, its performance hotspots. There are several different manners by which we can profile application run time in MATLAB, and different situations may dictate different tools.

Chapters 3 through 11 discuss specific speedup techniques that can be used in MATLAB:

- Chapter 3 explains standard techniques adapted from non-MATLAB programming languages.
- Chapter 4 discusses techniques that are unique to MATLAB code.
- Chapter 5 discusses implicit parallelization, with indexing and vectorization.
- Chapters 6 and 7 discuss explicit parallelization using a variety of means (CPU, GPU, and multi-threading).
- Chapter 8 discusses techniques for using compiled (binary) code.
- Chapter 9 discusses specific techniques that are memory-related. The nontrivial relationship between memory and performance is explained, and a variety of tuning techniques are presented in light of these explanations.
- Chapter 10 discusses techniques related to graphics, GUI, and user interaction.
- Chapter 11 concludes the list of specific tuning techniques with a discussion of techniques related to I/O, particularly reading and writing files.

Chapters 3 through 11 are intended for use as a random-access reference. The sections and techniques can typically be used independently of each other. You can directly use any section or technique, without reading or using any other.

Appendix A presents online and offline resources that expand the information presented in the text and enable further research. Appendix B concludes the text by providing a non-comprehensive general checklist for performance tuning.

Throughout the text, references are provided to enable interested readers to expand their knowledge of specific issues. Footnotes are used to clarify some points and to provide cross-references to other sections within this book; endnotes are used to provide references to related online resources. Most online references are provided in both full and shortened format, to enable easy usage when transcribed from hardcopy.

Conventions Used in This Book

The following special text formatting conventions are used within this book:

- `Fixed-width` font is used for MATLAB code segments. The Command-Line prompt (>>) is provided only where it would help to distinguish between user-entered text and MATLAB's response in the console:

```
>> version
ans =
8.3.0.532 (R2014a)
```

In other places, the console output is indicated using an arrow sign:

```
tic, pause(2), toc
⇨ Elapsed time is 2.001925 seconds.
```

- **Regular bold** font is used for object property names (e.g., **UserData**), as well as for occasional emphasis.
- ***Bold italic*** font is used for MATLAB function names (e.g., ***max*** or ***ismember***)
- *Regular italic* font is used for file names (e.g., *data.mat*), utility names (e.g., *grep*), introduction of new terms, as well as for occasional emphasis.

The duration terms "minutes", "seconds", and "milliseconds" are used extensively throughout the text and are usually shortened to "min", "s", and "ms", respectively (e.g., 3 min, 5 s or 45 ms).

Icons are sometimes placed next to the text, as follows:

- The lightning icon[3] indicates a suggestion with potentially high impact on the program's performance. These suggestions should typically be considered first, before trying other alternatives.

- The warning icon indicates a suggestion that relies on undocumented or unsupported features that may not be available in future MATLAB releases and may not work correctly (or at all) on some platforms, releases, or situations. Use of such suggestions should be done only after careful testing and at the user's own risk. Neither MathWorks nor this book's author or publisher can take any responsibility for possible consequences due to using unsupported functionality. It is the author's explicit suggestion that such features should be considered last, only after all the other (supported) venues have been tried.

Acknowledgments

I owe a debt of gratitude to several MATLAB experts who have helped me prepare this manuscript. A few experts were most gracious to provide significant contributions in their field of expertise: Pavel Emeliyanenko (parallelization); James Tursa (MEX), and Igal Yaroslavski (FPGA). I received invaluable assistance from MathWorks, both in the Books Program (Naomi Fernandes) and R&D (Michelle Hirsch, Ken Atwell, Gaurav Sharma, Pat

Quillen, and others).* Systematics Ltd. were very helpful: Michael Donnefeld helped me wade through MATLAB Coder issues, Igal provided the FPGA section, and David Gochman assisted behind-the-scenes. John D'Errico and Bruno Luong are MATLAB giants—I have learned countless techniques from them over the years. Jim Hokanson, Bill McKeeman, Malcolm Lidierth, Mike Croucher, Eric Sampson, Kadin Tseng, and Oliver Woodford reviewed and provided valuable feedback. At CRC Press I found a truly caring publisher: Bob Stern and Bob Ross for helping with issues large and small, always with patience, understanding, and a kind word; and Kyle Meyer and Syed Shajahan for careful editing.

This book is much better thanks to the contributions by these people, and yet I take responsibility for any errors, inaccuracies, and omissions that might have inadvertently entered the text. In such a large project, as in any large program, some "bugs" have surely escaped scrutiny. I hope that you accept them with the understanding that such work is never perfect. I have tried to generalize my findings for improved performance, and yet it is quite possible that the situation may be different or even diametrically opposite under a different set of parameters such as hardware, OS, MATLAB release, data size, data type, or any of myriad other possible factors. Readers should therefore not rely on the suggestions within the text before carefully testing them on their specific system. Neither the author nor publisher can accept any responsibility for possible consequences due to this book. Please report any issue that you discover. My direct email is altmany (at) gmail.com. A detailed list of the major errata will be posted on http://UndocumentedMatlab.com/books/matlab-performance.

Finally, this book is devoted to my family. I said it in my first book and find no better words to say it again—this book would never have seen the light of day without your loving support and understanding. I owe you more than words can express.

* MathWorks employees are affectionately called *MathWorkers* by the MATLAB community. I use this term in this book to refer to MathWorks engineers, either past or current, who contributed some useful utility, suggestion, or insight.

Author

Yair Altman, author of the extremely popular UndocumentedMatlab.com website, is well respected by the MATLAB® community as a leading expert on advanced MATLAB programming.

Yair's first book, *Undocumented Secrets of MATLAB-Java Programming*, was published in 2011 to rave reviews and became the standard textbook on the subject. His many years of public contribution on MATLAB performance, plus a multitude of useful tips never before published, are now available in this highly readable volume.

Yair holds a BSc in physics and an MSc in computer science, both with high honors. Yair has over 20 years of professional software development experience at various levels of organizational responsibility, from programmer to VP R&D. He has developed systems using two dozen programming languages, on a dozen different platforms, half a dozen databases, and countless MATLAB releases.

Yair became an independent MATLAB consultant several years ago, and has never looked back. He currently assists clients world-wide in various MATLAB-related aspects: consulting, training, and programming.

Yair can be reached at altmany@gmail.com.

Contents at a Glance