Preface

MATLAB® is currently the dominant language of technical computing with approximately one million users worldwide, many of whom can benefit from the increased power offered by widely available multicore processors and multinode computing clusters. MATLAB is also an ideal environment for learning about parallel computing, allowing the user to focus on parallel algorithms instead of the details of the implementation. The succinctness of MATLAB allows many specific examples to be presented to illustrate parallel programming concepts.

The subject of this book is parallel programming in MATLAB and is hopefully the first of many books on this topic, as there are now a wide variety of parallel MATLAB libraries [Choy 2004] for users to choose from. I am fortunate to have been involved in the development of two of these libraries [Kepner 2004, Kepner 2003]. The most widely used of these libraries include pMatlab (developed by MIT Lincoln Laboratory), the Parallel Computing Toolbox (developed by The MathWorks, Inc.), and StarP (developed at MIT, UCSB, and Interactive Supercomputing, Inc.). All of these libraries provide direct support for parallel computing using distributed arrays and other parallel programming models. The specific examples in this book are written using the freely available pMatlab library. pMatlab has the advantage of running either standalone or on top of the aforementioned parallel MATLAB libraries. So all the examples in the book can be run in any of the parallel MATLAB environments. Fortunately, the concepts illustrated in the book are independent of the underlying implementation and valid for any particular parallel programming syntax the user may prefer to use. The software along with installation instructions can be found at the book website:

http://www.siam.org/KepnerBook

The expressive power of MATLAB allows us to concentrate on the techniques for creating parallel programs that run well (as opposed to the syntactic mechanics of writing parallel programs). Our primary focus is on the designing, coding, debugging, and testing techniques required to quickly produce well-performing parallel programs in a matter of hours instead of weeks or months (see Figure 1). These techniques have been developed over the years through thousands of one-on-one interactions with hundreds of parallel MATLAB users.

A general familiarity with MATLAB is assumed (see [Higham & Higham 2005] and [Moler 2004] for an introduction to MATLAB). The target audience of the book

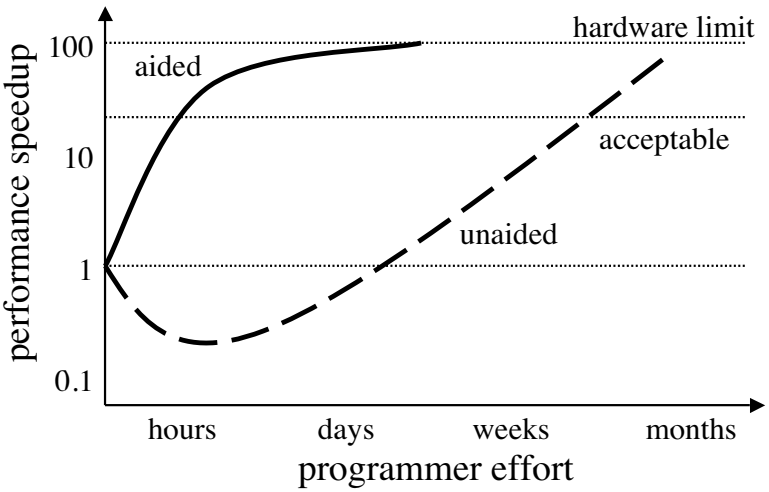

Figure 1. Performance versus effort.
Depiction of the performance achieved versus the user effort invested for a typical parallel MATLAB program. The bottom curve shows the effort required for an unaided user. The top curve shows the effort required for a user aided with expert assistance.

is anyone who needs to adapt their serial MATLAB program to a parallel environment. In addition, this book is suitable as either the primary book in a parallel computing class or as a supplementary text in a numerical computing class or a computer science algorithms class. Ideas are presented using a "hands-on" approach with numerous example programs. Wherever possible, the examples are drawn from widely known and well-documented parallel benchmark codes that have already been identified as representing many applications (although the connection to any particular application may require examining the references). For historical reasons, most of the examples are drawn from the technical computing domain, but an understanding of numerical methods is *not* required in order to use the examples. They are simply convenient and well-documented examples of different types of parallel programming.

The book is organized around two central concepts: the core programming process (i.e., design, code, debug, and test) and the core parallel programming models (i.e., distributed arrays, manager/worker, and message passing) [Lusk 2004]. The distributed array programming model will be the baseline programming model used throughout the book. Distributed arrays are easiest to understand, require the least amount of code to use, and are well-matched to the array nature of MATLAB. Distributed arrays allow very complicated communication patterns to be illustrated far more simply than with a message passing approach and perform well on both multicore and multinode parallel computers. In addition, distributed arrays naturally illustrate the core parallel design concepts of concurrency and locality. Distributed

arrays are sufficient for 90% of parallel applications, but there are instances where other programming models are optimal and these are discussed where appropriate. The ultimate goal of this book is teach the reader to "think [distributed] matrices, not messages" [Moler 2005].

Throughout the book the approach is to first present concrete examples and then discuss in detail the more general parallel programming concepts these examples illustrate. The book aims to bring these ideas to bear on the specific challenge of writing parallel programs in MATLAB.

To accommodate the different types of readers, the book is organized into three parts. Part I (Chapters 1–3) provides a general conceptual overview of the key programming concepts illustrated by specific examples. Part II (Chapters 4–6) focuses on the analysis techniques of effective parallel programming. Part III (Chapters 7–11) consists of specific case studies. Parts I, II, and III can be treated independently and may be used as modules in a larger course. Finally, in recognition of the severe time constraints of professional users, each chapter is mostly self-contained and key terms are redefined as needed. Each chapter has a short summary and references within that chapter are listed at the end of the chapter. This arrangement allows the professional user to pick up and use any particular chapter as needed.

Chapter 1 begins by introducing some notation and the basic parallel programming interfaces used throughout the rest of the book. Chapter 2 provides a rapid introduction to creating and running simple parallel programs. Chapter 2 is meant to give readers a taste of the ease of use that parallel MATLAB provides. Chapter 3 focuses on a more complex example and highlights interacting with distributed arrays. Chapter 4 is a broad overview of the field of parallel programming, exposing the key principles that will be covered in more detail in the rest of the book. Chapter 4 uses a more sophisticated example than that used in Chapter 3, exposing the boundaries of the different parallel programming models. In addition, Chapter 4 introduces several key questions that must be dealt with in parallel programming:

Design: when to use parallel processing, concurrency versus locality, how to predict parallel performance, which parallel programming model to use: distributed arrays, client/server, or message passing.

Code: how and where to use parallel code, how to write scalable code (e.g., scalable file I/O), good coding style.

Debug: what the techniques are for going from a serial to a fully parallel execution, what kind of errors to check for at each stage along the way.

Test: how to measure performance achieved and compare with what is predicted.

Chapter 5 introduces the theory, algorithmic notation, and an "under the hood" view of distributed array programming. Chapter 6 discusses metrics for evaluating performance and coding of a parallel program. Chapter 7 is a selected survey of parallel application analysis techniques, with a particular emphasis on how the examples used in the book relate to many wider application domains. Chapters 8–11 use a set of well-studied examples drawn from the HPC Challenge benchmark

suite (see http://www.hpcchallenge.org) to show detailed solutions to the challenges raised by parallel design, coding, debugging, and testing.

Throughout this book we draw upon numerous classical parallel programming examples that have been well studied and characterized in the literature. It is my hope that in seeing these examples worked out the reader will come to appreciate the elegance of parallel programming in MATLAB as much as I have.

References

[Choy 2004] Ron Choy, Parallel Matlab survey, 2004, http://supertech.lcs.mit.edu/~cly/survey.html

[Higham & Higham 2005] Desmond J. Higham and Nicholas J. Higham, MATLAB Guide, Second Edition, SIAM, Philadelphia, 2005.

[Kepner 2004] Jeremy Kepner and Stan Ahalt, MatlabMPI, Journal of Parallel and Distributed Computing, Vol. 64, No. 8, pp. 997–1005, 2004.

[Kepner 2003] Jeremy Kepner and Nadya Travinin, Parallel Matlab: The Next Generation, Seventh Annual High Performance Embedded Computing Workshop (HPEC 2003), September 23–25, 2003, MIT Lincoln Laboratory, Lexington, MA, http://www.ll.mit.edu/HPEC/agenda03.htm

[Lusk 2004] Ewing Lusk and Marc Snir, Parallel Programming Models, DARPA HPCS Productivity Team Workshop, January 13–14, 2004, Marina Del Ray, CA.

[Moler 2004] Cleve Moler, Numerical Computing with MATLAB, SIAM, Philadelphia, 2004.

[Moler 2005] Cleve Moler, Householder Meeting on Numerical Linear Algebra, Champion, PA, May 23–27, 2005.