Preface

As the title suggests, this book is both an introduction to MATLAB programming and to the computational side of science and engineering. We target college freshman intending to major in engineering (including computer science) or a natural science (including mathematics). Given the quantitative abilities of this group of students, we do not shy away from trigonometry and elementary notions of approximation as seen in Calculus I. Indeed, it is against the grain of liberal education not to intermingle introductory programming with continuous mathematics if the student clientele is capable of handling the mix. Liberal education is all about acquiring an appreciation for different modes of thought. Why squander the opportunity to contrast digital thinking with continuous thinking?

Our approach is simple. Each section begins with the posing of a problem that points to some larger computational story. The solution is carefully derived and along the way we introduce whatever new MATLAB is required. This is followed by a brief "talking point" that emphasizes some aspect of the larger story. This pattern resonates with our belief that a first course in programming should be taught through examples. Every section culminates in the production of a working MATLAB script and (usually) a few MATLAB functions. The section exercises include straight forward "**M**-problems" that focus on the developed codes and whatever new MATLAB is developed. More involved "**P**-problems" are designed to reinforce the section's computational message.

We use the MATLAB environment because of its friendliness to the first-time programmer and because it supports the idea of playing with computational ideas through experimentation. This is central to the development of computational intuition.

Playing with programs builds computational intuition.

Intuition is a sense of direction no different from the sense of direction that enables you to find your way around an old childhood neighborhood without a map. The key is that you have been there before. If intuition is a sense of direction, then computational intuition is a sense of computational direction. Those who have it will be able to find their way around science and engineering in the 21st century. Navigation requires five keen senses. Through examples and problems we aim to

1. *Develop eyes for the geometric.* The ability to visualize is very important to the computational scientist. Of course, computer graphics plays a tremendous role here, but the visualization tools that it offers do not obviate the need to reason in geometric terms. It is critical to be familiar with sines and cosines, polygons and polyhedra, metrics and proximity, etc.

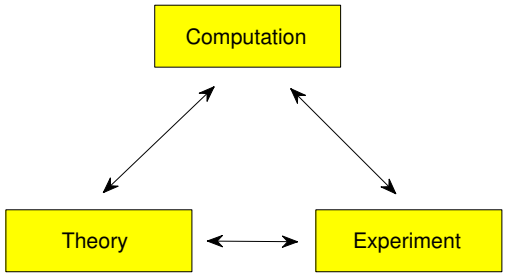
2. *Develop an ear that can hear the "combinatoric explosion."* Many design and optimization problems involve huge search spaces with an exponential number of possibilities. It is important to anticipate this complexity and to have the wherewithal to handle it with intelligent heuristics.

3. *Develop a taste for the random.* Science and engineering is populated with processes that have a random component. Having a sense of probability and the ability to gather and interpret statistics with the computer is vital.

4. *Develop a nose for dimension.* Simulation is much more computationally intensive in three dimensions than in two dimensions – a hard fact of life that is staring many computational scientists right in the face. An accurate impression of how computers assist in the understanding of the physical world requires an appreciation of this point. Moreover, being able to think at the array level is essential for effective, high-performance computing.

5. *Develop a touch for what is finite, inexact, and approximate.* Rounding errors attend floating-point arithmetic, computer displays are granular, analytic derivatives are approximated with divided differences, a polynomial is used in lieu of the sine function, and the data acquired in a lab may only be correct to three significant digits. Life in computational science is like this and the practitioner must be solid enough to face such uncertainties. Steady footwork is required on the balance beam that separates the continuous and the discrete.

While the development of these five senses is an explicit priority, our overarching ambition is to communicate the excitement of computing together with an appreciation for its constraints and its connections to other methodologies. The interplay between computing, theory, and experimentation is particularly important:



Each vertex represents a style of research and provides a window through which we can look at science and engineering in the large. The vibrancy of what we see inside the triangle depends upon the ideas that flow around its edges. A good theory couched in the language of mathematics may be realized in the form of a computer program, perhaps just to affirm its correctness. Running the program results in a

simulation that may suggest a physical experiment. The experiment in turn may reveal a missed parameter in the underlying mathematical model, and around we go again.

There are also interesting dynamics in the other direction. A physical experiment may be restricted in scope for reasons of budget or safety, so the scene shifts to computer simulation. The act of writing the program to perform the simulation will most likely have a clarifying influence, prompting some new mathematical pursuit. Innovative models are discovered, leading to a modification of the initial set of experiments, and so forth.

In thinking about these critical interactions we are reminded of the great mathematical scientist Richard Hamming who stated in the 1960's that "the purpose of computing is insight, not numbers." We are in obvious agreement with this point of view. The takeaway message from a first programming course should be "Insight Through Computing" instead of just "Output Through Computing." The next generation of computational scientists and engineers needs to think broadly and creatively and we hope that our book is a contribution in that direction.

Acknowledgements

This book is derived from many years of experience teaching CS 100M (now CS 1112) at Cornell University. We are indebted to the many graduate student teaching assistants who, through their hard work, have given us the time to refine our course notes and to develop interesting assignments. In particular, we would like to thank Mr. Tim Condon for co-authoring Appendix C.

More generally, we are fortunate to have our academic home defined by two of Cornell's great academic units: the College of Engineering and the Faculty of Computing and Information Science. Location is everything if you want to be energized by both colleagues and students. We have the best.

Daisy Fan
Charlie Van Loan
Ithaca, New York