Preface

Computing instruments were developed to facilitate fast calculations, especially in computational science and engineering applications. The fact that this is not just a matter of building a hardware device and its system software, was already hinted to by Charles Babbage, when he wrote in the mid-nineteenth century, *As soon as an Analytical Engine exists, it will necessarily guide the future course of science. Whenever any result is sought by its aid, the question will then arise—By what course of calculation can these results be arrived at by the machine in the shortest time?* [1]. This question points to one of the principal challenges for parallel computing. In fact, in the aforementioned reference, Babbage did consider the advantage of parallel processing and the perfect speedup that could be obtained when adding numbers if no carries were generated. He wrote *If this could be accomplished it would render additions and subtractions with numbers having ten, twenty, fifty or any number of figures as rapid as those operations are with single figures*. He was also well aware of the limitations, in this case the dependencies caused by the carries. A little more than half a century after Babbage, in 1922, an extraordinary idea was sketched by Lewis Fry Richardson. In his treatise *Weather Prediction by Numerical Process* he described his "forecast-factory" fantasy to speed up calculations by means of parallel processing performed by humans [3, Chap. 11, p. 219]. Following the development of the first electronic computer, in the early 1950s, scientists and engineers proposed that one way to achieve higher performance was to build a computing platform consisting of many interconnected von Neumann uniprocessors that can cooperate in handling what were the large computational problems of that era. This idea appeared simple and natural, and quickly attracted the attention of university-, government-, and industrial-research laboratories. Forty years after Richardson's treatise, the designers of the first parallel computer prototype ever built, introduced their design in 1962 as follows: *The Simultaneous Operation Linked Ordinal Modular Network (SOLOMON), a parallel network computer, is a new system involving the interconnections and programming, under the supervision of a central control unit, of many identical processing elements (as few or as many as a given problem requires), in an arrangement that can simulate directly the problem being solved.* It is remarkable how this

introductory paragraph underlines the generality and adaptive character of the design, despite the fact that neither the prototype nor subsequent designs went as far. These authors stated further that this architecture shows great promise in aiding progress in certain critical applications that rely on common mathematical denominators that are dominated by matrix computations.

Soon after that, the field of parallel processing came into existence starting with the development of the ILLIAC-IV at the University of Illinois at Urbana-Champaign led by Daniel L. Slotnick (1931–1985), who was one of the principal designers of the SOLOMON computer. The design and building of parallel computing platforms, together with developing the underlying system software as well as the associated numerical libraries, emerged as important research topics. Now, four decades after the introduction of the ILLIAC-IV, parallel computing resources ranging from multicore systems (which are found in most modern desktops and laptops) to massively parallel platforms are within easy reach of most computational scientists and engineers. In essence, parallel computing has evolved from an exotic technology to a widely available commodity. Harnessing this power to the maximum level possible, however, remains the subject of ongoing research efforts.

Massively parallel computing platforms now consist of thousands of nodes cooperating via sophisticated interconnection networks with several layers of hierarchical memories. Each node in such platforms is often a multicore architecture. Peak performance of these platforms has reached the persetascale level, in terms of the number of floating point operations completed in one second, and will soon reach the exascale level. These rapid hardware technological advances, however, have not been matched by system or application software developments. Since the late 1960s different parallel architectures have come and gone in a relatively short time resulting in lack of stable and sustainable parallel software infrastructure. In fact, present day researchers involved in the design of parallel algorithms and development of system software for a given parallel architecture often rediscover work that has been done by others decades earlier. Such lack of stability in parallel software and algorithm development has been pointed out by George Cybenko and David Kuck as early as 1992 in [2].

Libraries of efficient parallel algorithms and their underlying kernels are needed for enhancing the realizable performance of various computational science and engineering (CSE) applications on current multicore and petascale computing platforms. Developing robust parallel algorithms, together with their theoretical underpinnings is the focus of this book. More specifically, we focus exclusively on those algorithms relevant to dense and sparse matrix computations which govern the performance of many CSE applications. The important role of matrix computations was recognized in the early days of digital computers. In fact, after the introduction of the Automatic Computing Engine (ACE), Alan Turing included solving linear systems and matrix multiplication as two of the computational challenges for this computing platform. Also, in what must be one of the first references to sparse and structured matrix computations, he observed that even though the storage capacities available then could not handle dense linear systems

of order larger than 50, in practice one can handle much larger systems: *The majority of problems have very degenerate matrices and we do not need to store anything like as much as ... since the coefficients in these equations are very systematic and mostly zero.* The computational challenges we face today are certainly different in scale than those above but they are surprisingly similar in their dependence on matrix computations and numerical linear algebra. In the early 1980s, during building the experimental parallel computing platform "Cedar", led by David Kuck, at the University of Illinois at Urbana-Champaign, a table was compiled that identifies the common computational bottlenecks of major science and engineering applications, and the parallel algorithms that need to be designed, together with their underlying kernels, in order to achieve high performance. Among the algorithms listed, matrix computations are the most prominent. A similar list was created by UC Berkeley in 2009. Among Berkeley's 13 parallel algorithmic methods that capture patterns of computation and communication, which are called "dwarfs", the top two are matrix computation-based. Not only are matrix computations, and especially sparse matrix computations, essential in advancing science and engineering disciplines such as computational mechanics, electromagnetics, nanoelectronics among others, but they are also essential for manipulation of the large graphs that arise in social networks, sensor networks, data mining, and machine learning just to list a few. Thus, we conclude that realizing high performance in dense and sparse matrix computations on parallel computing platforms is central to many applications and hence justify our focus.

Our goal in this book is therefore to provide researchers and practitioners with the basic principles necessary to design efficient parallel algorithms for dense and sparse matrix computations. In fact, for each fundamental matrix computation problem such as solving banded linear systems, for example, we present a family of algorithms. The "optimal" choice of a member of this family will depend on the linear system and the architecture of the parallel computing platform under consideration. Clearly, however, executing a computation on a parallel platform requires the combination of many steps ranging from: (i) the search for an "optimal" parallel algorithm that minimizes the required arithmetic operations, memory references and interprocessor communications, to (ii) its implementation on the underlying platform. The latter step depends on the specific architectural characteristics of the parallel computing platform. Since these architectural characteristics are still evolving rapidly, we will refrain in this book from exposing fine implementation details for each parallel algorithm. Rather, we focus on algorithm robustness and opportunities for parallelism in general. In other words, even though our approach is geared towards numerically reliable algorithms that lend themselves to practical implementation on parallel computing platforms that are currently available, we will also present classes of algorithms that expose the theoretical limitations of parallelism if one were not constrained by the number of cores/processors, or the cost of memory references or interprocessor communications.

In summary, this book is intended to be both a research monograph as well as an advanced graduate textbook for a course dealing with parallel algorithms in matrix computations or numerical linear algebra. It is assumed that the reader has general, but not extensive, knowledge of: numerical linear algebra, parallel architectures, and parallel programming paradigms. This book consists of four parts for a total of 13 chapters. Part I is an introduction to parallel programming paradigms and primitives for dense and sparse matrix computations. Part II is devoted to dense matrix computations such as solving linear systems, linear least squares and algebraic eigenvalue problems. Part II also deals with parallel algorithms for special matrices such as banded, Vandermonde, Toeplitz, and block Toeplitz. Part III deals with sparse matrix computations: (a) iterative parallel linear system solvers with emphasis on scalable preconditioners, (b) schemes for obtaining few of the extreme or interior eigenpairs of symmetric eigenvalue problems, (c) schemes for obtaining few of the singular triplets. Finally, Part IV discusses parallel algorithms for computing matrix functions and the matrix pseudospectrum.

Acknowledgments

We wish to thank all of our current and previous collaborators who have been, directly or indirectly, involved with topics discussed in this book. We thank especially: Guy-Antoine Atenekeng-Kahou, Costas Bekas, Michael Berry, Olivier Bertrand, Randy Bramley, Daniela Calvetti, Peter Cappello, Philippe Chartier, Michel Crouzeix, George Cybenko, Ömer Eğecioğlu, Jocelyne Erhel, Roland Freund, Kyle Gallivan, Ananth Grama, Joseph Grcar, Elias Houstis, William Jalby, Vassilis Kalantzis, Emmanuel Kamgnia, Alicia Klinvex, Çetin Koç, Efi Kokiopoulou, George Kollias, Erricos Kontoghiorghes, Alex Kouris, Ioannis Koutis, David Kuck, Jacques Lenfant, Murat Manguoğlu, Dani Mezher, Carl Christian Mikkelsen, Maxim Naumov, Antonio Navarra, Louis Bernard Nguenang, Nikos Nikoloutsakos, David Padua, Eric Polizzi, Lothar Reichel, Yousef Saad, Miloud Sadkane, Vivek Sarin, Olaf Schenk, Roger Blaise Sidje, Valeria Simoncini, Aleksandros Sobczyk, Danny Sorensen, Andreas Stathopoulos, Daniel Szyld, Maurice Tchuente, Tayfun Tezduyar, John Tsitsiklis, Marian Vajteršic, Panayot Vassilevski, Ioannis Venetis, Brigitte Vital, Harry Wijshoff, Christos Zaroliagis, Dimitris Zeimpekis, Zahari Zlatev, and Yao Zhu. Any errors and omissions, of course, are entirely our responsibility.

In addition we wish to express our gratitude to Yousuff Hussaini who encouraged us to have our book published by Springer, to Connie Ermel who typed a major part of the first draft, to Eugenia-Maria Kontopoulou for her help in preparing the index, and to our Springer contacts, Kirsten Theunissen and Aldo Rampioni. Finally, we would like to acknowledge the remarkable contributions of the late Gene Golub—a mentor and a friend—from whom we learned a lot about matrix computations. Further, we wish to pay our respect to the memory of our late collaborators and friends: Theodore Papatheodorou, and John Wisniewski.

Last, but not least, we would like to thank our families, especially our spouses, Aristoula, Elisabeth and Marilyn, for their patience during the time it took us to produce this book.

Patras — Efstratios Gallopoulos
Rennes — Bernard Philippe
West Lafayette — Ahmed H. Sameh
January 2015

References

1. Babbage, C.: Passages From the Life of a Philosopher. Longman, Green, Longman, Roberts & Green, London (1864)
2. Cybenko, G., Kuck, D.: Revolution or Evolution, IEEE Spectrum, **29**(9), 39–41 (1992)
3. Richardson, L.F.: Weather Prediction by Numerical Process. Cambridge University Press, Cambridge (1922). (Reprinted by Dover Publications, 1965)

http://www.springer.com/978-94-017-7187-0

Parallelism in Matrix Computations
Gallopoulos, E.; Philippe, B.; Sameh, A.H.
2016, XXX, 473 p. 58 illus., Hardcover
ISBN: 978-94-017-7187-0