Are Magnus Bruaset
Aslak Tveito

# Numerical Solution of Partial Differential Equations on Parallel Computers

September 8, 2005

**Springer**
*Berlin Heidelberg New York*
*Hong Kong London*
*Milan Paris Tokyo*

# Preface

Since the dawn of computing, the quest for a better understanding of Nature has been a driving force for technological development. Groundbreaking achievements by great scientists have paved the way from the abacus to the supercomputing power of today. When trying to replicate Nature in the computer's silicon test tube, there is need for precise and computable process descriptions. The scientific fields of Mathematics and Physics provide a powerful vehicle for such descriptions in terms of Partial Differential Equations (PDEs). Formulated as such equations, physical laws can become subject to computational and analytical studies. In the computational setting, the equations can be discretized for efficient solution on a computer, leading to valuable tools for simulation of natural and man-made processes. Numerical solution of PDE-based mathematical models has been an important research topic over centuries, and will remain so for centuries to come.

In the context of computer-based simulations, the quality of the computed results is directly connected to the model's complexity and the number of data points used for the computations. Therefore, computational scientists tend to fill even the largest and most powerful computers they can get access to, either by increasing the size of the data sets, or by introducing new model terms that make the simulations more realistic, or a combination of both. Today, many important simulation problems can not be solved by one single computer, but calls for *parallel computing*. Whether being a dedicated multi-processor supercomputer or a loosely coupled cluster of office workstations, the concept of parallelism offers increased data storage and increased computing power. In theory, one gets access to the grand total of the resources offered by the individual units that make up the multi-processor environment. In practice, things are more complicated, and the need for data communication between the different computational units consumes parts of the theoretical gain of power.

Summing up the bits and pieces that go into a large-scale parallel computation, there are aspects of hardware, system software, communication protocols, memory management, and solution algorithms that have to be addressed. However, over time efficient ways of addressing these issues have emerged, better software tools have become available, and the cost of hardware has fallen considerably. Today, computational clusters made from commodity parts can be set up within the budget of a typ-

ical research department, either as a turn-key solution or as a do-it-yourself project. Supercomputing has become affordable and accessible.

*About this book*

This book addresses the major topics involved in numerical simulations on parallel computers, where the underlying mathematical models are formulated in terms of PDEs. Most of the chapters dealing with the technological components of parallel computing are written in a survey style and will provide a comprehensive, but still readable, introduction for students and researchers. Other chapters are more specialized, for instance focusing on a specific application that can demonstrate practical problems and solutions associated with parallel computations. As editors we are proud to put together a volume of high-quality and useful contributions, written by internationally acknowledged experts on high-performance computing.

The first part of the book addresses fundamental parts of parallel computing in terms of hardware and system software. These issues are vital to all types of parallel computing, not only in the context of numerical solution of PDEs. To start with, Ricky Kendall and co-authors discuss the programming models that are most commonly used for parallel applications, in environments ranging from a simple departmental cluster of workstations to some of the most powerful computers available today. Their discussion covers models for message passing and shared memory programming, as well as some future programming models. In a closely related chapter, Jim Teresco et al. look at how data should be partitioned between the processors in a parallel computing environment, such that the computational resources are utilized as efficient as possible. In a similar spirit, the contribution by Martin Rumpf and Robert Strzodka also aims at improved utilization of the available computational resources. However, their approach is somewhat unconventional, looking at ways to benefit from the considerable power available in graphics processors, not only for visualization purposes but also for numerical PDE solvers. Given the low cost and easy access of such commodity processors, one might imagine future cluster solutions with really impressive price-performance ratios.

Once the computational infrastructure is in place, one should concentrate on how the PDE problems can be solved in an efficient manner. This is the topic of the second part of the book, which is dedicated to parallel algorithms that are vital to numerical PDE solution. Luca Formaggia and co-authors present parallel domain decomposition methods. In particular, they give an overview of algebraic domain decomposition techniques, and introduce sophisticated preconditioners based on a multilevel approximative Schur complement system and a Schwarz-type decomposition, respectively. As Schwarz-type methods call for a coarse level correction, the paper also proposes a strategy for constructing coarse operators directly from the algebraic problem formulation, thereby handling unstructured meshes for which a coarse grid can be difficult to define. Complementing this multilevel approach, Frank Hülsemann et al. discuss how another important family of very efficient PDE solvers, geometric multigrid, can be implemented on parallel computers. Like domain decomposition methods, multigrid algorithms are potentially capable of being order-optimal such

that the solution time scales linearly with the number of unknowns. However, this paper demonstrates that in order to maintain high computational performance the construction of a parallel multigrid solver is certainly problem-dependent. In the following chapter, Ulrike Meier Yang addresses parallel algebraic multigrid methods. In contrast to the geometric multigrid variants, these algorithms work only on the algebraic system arising from the discretization of the PDE, rather than on a multiresolution discretization of the computational domain. Ending the section on parallel algorithms, Nikos Chrisochoides surveys methods for parallel mesh generation. Meshing procedures are an important part of the discretization of a PDE, either used as a preprocessing step prior to the solution phase, or in case of a changing geometry, as repeated steps in course of the simulation. This contribution concludes that it is possible to develop parallel meshing software using off-the-shelf sequential codes as building blocks without sacrificing the quality of the constructed mesh.

Making advanced algorithms work in practice calls for development of sophisticated software. This is especially important in the context of parallel computing, as the complexity of the software development tends to be significantly higher than for its sequential counterparts. For this reason, it is desirable to have access to a wide range of software tools that can help make parallel computing accessible. One way of addressing this need is to supply high-quality software libraries that provide parallel computing power to the application developer, straight out of the box. The *hypre* library presented by Robert D. Falgout et al. does exactly this by offering parallel high-performance preconditioners. Their paper concentrates on the conceptual interfaces in this package, how these are implemented for parallel computers, and how they are used in applications. As an alternative, or complement, to the library approach, one might look for programming languages that tries to ease the process of parallel coding. In general, this is a quite open issue, but Xing Cai and Hans Petter Langtangen contribute to this discussion by considering whether the high-level language Python can be used to develop efficient parallel PDE solvers. They address this topic from two directions, looking at the performance of parallel PDE solvers mainly based on Python code and native data structures, and through use of Python to parallelize existing sequential PDE solvers written in a compiled language like FORTRAN, C or C++. The latter approach also opens for the possibility of combining different codes in order to address a multi-model or multiphysics problem. This is exactly the concern of Lois Curfman McInnes and her co-authors when they discuss the use of the Common Component Architecture (CCA) for parallel PDE-based simulations. Their paper gives an introduction to CCA and highlights several parallel applications for which this component technology is used, ranging from climate modeling to simulation of accidental fires and explosions.

To communicate experiences gained from work on some complete simulators, selected parallel applications are discussed in the latter part of the book. Xing Cai and Glenn Terje Lines present work on a full-scale parallel simulation of the electrophysiology of the human heart. This is a computationally challenging problem, which due to a multiscale nature requires a large amount of unknowns that have to be resolved for small time steps. It can be argued that full-scale simulations of this problem can not be done without parallel computers. Another challenging geody-

namics problem, modeling the magma genesis in subduction zones, is discussed by Matthew G. Knepley et al. They have ported an existing geodynamics code to use PETSc, thereby making it parallel and extending its functionality. Simulations performed with the resulting application confirms physical observations of the thermal properties in subduction zones, which until recently were not predicted by computations. Finally, in the last chapter of the book, Carolin Körner et al. present parallel Lattice Boltzmann Methods (LBMs) that are applicable to problems in Computational Fluid Dynamics. Although not being a PDE-based model, the LBM approach can be an attractive alternative, especially in terms of computational efficiency. The power of the method is demonstrated through computation of 3D free surface flow, as in the interaction and growing of gas bubbles in a melt.

*Acknowledgements*

We wish to thank all the chapter authors, who have written very informative and thorough contributions that we think will serve the computational community well. Their enthusiasm has been crucial for the quality of the resulting book.

Moreover, we wish to express our gratitude to all reviewers, who have put time and energy into this project. Their expert advice on the individual papers has been useful to editors and contributors alike. We are also indebted to Dr. Martin Peters at Springer-Verlag for many interesting and useful discussions, and for encouraging the publication of this volume.

Simula Research Laboratory
Fornebu, September 2005

*Are Magnus Bruaset*
*Aslak Tveito*

# Contents