Preface

Hans Petter Langtangen

Simula Research Laboratory,
P.O. Box 134,
NO-1325 Lysaker, Norway

Department of Informatics,
University of Oslo,
P.O. Box 1080, Blindern
N-0316 Oslo, Norway
E-mail: hpl@simula.no

Michael Thuné

Department of Information Technology,
Uppsala University,
P.O. Box 337,
SE-751 05 Uppsala, Sweden
E-mail: michael.thune@it.uu.se

Biographical notes: Hans Petter Langtangen is a Professor of Mathematical Modelling at the Department of Informatics, University of Oslo, and a director of the Centre for Biomedical Computing at Simula Research Laboratory. His research concerns numerical methods for partial differential equations, scientific software, and applications to problems in mechanics. He is a principal developer of Diffpack, a software framework for solving partial differential equations, and the author of several books on mathematical modelling OG scientific programming.

Michael Thuné is a Professor of Scientific Computing at the Department of Information Technology, Uppsala University. While his work in early years concerned development of numerical methods, he has over a period of almost two decades built a research group performing fundamental research on how to design software frameworks that significantly simplify the implementation of numerical methods for solving partial differential equations on parallel computers.

Software is a crucial prerequisite for Computational Science and Engineering (CSE). However, while other key ingredients such as mathematical modelling and numerical analysis are regarded as research areas, it is not as widely recognised that the construction of software for CSE gives rise to highly important research issues.

"Software Issues in Computational Science and Engineering" was the theme of a workshop at Uppsala University, Sweden, 15–16 August, 2007. The event was organised jointly by Uppsala University and Simula Research Laboratory and was held in conjunction with the annual meeting of IFIP Working Group 2.5 ('Numerical Software'). See http://www.it.uu.se/research/conf/SCSE07 for more information.

The workshop addressed challenging research issues regarding software for numerical computations. Ideally, adaptation to new applications should be flexible, and extension to incorporate new numerical techniques straightforward. At the same time the software should execute extremely efficiently on various high-performance computing platforms. Accuracy and robustness are other key features. Complicated scientific software should also be easy to use to solve scientific or engineering problems. The overall challenge is to find ways to construct numerical software so that all these different goals are met simultaneously. The workshop provided a forum for researchers to present recent advances in this area.

This special issue of IJCSE contains papers based on some of the material presented at the workshop. The contributions can broadly be grouped into three themes: Software Engineering/Architecture; Implementation for Efficient Execution of Software; and Automatic Software Construction.

The Software Engineering/Architecture theme is addressed in the first three contributions. One key aspect of software engineering is to design software architectures that reduce the costs for implementation, modification, and maintenance of software. In this context the concept of 'software framework' is central.

One characteristic feature of a software framework is that it should support variability, to allow for construction of a variety of specific programs. The aspects that can

be varied within the framework are known as its 'variation points'. The contribution by Haveraaen and Friis is an argument for 'coordinate-free numerics' as an elegant way to achieve the variability required in a software framework for numerical solution of partial differential equations.

Software for Finite Element-based numerical solution of partial differential equations can be divided into a general-purpose library part and a problem-specific part. The paper by Alnaes et al. presents a unified framework to handle the interface and communication between the two parts. Such interfaces, if adopted as a standard, may ease the integration of different types of Finite Element software.

Due to the importance of software in CSE, training of future CSE specialists should include experience with realistic software engineering issues. At TU München, the developers of a CSE education has taken this into account by designing a project course where students develop software for scientific computing. The paper by Bader et al. gives a 'practice and experience' presentation of this initiative.

The theme of efficient implementation is addressed in the next three papers. A current trend is to enhance the performance of numerical software by carrying out all or parts of the computations on Graphics Processor Units (GPUs). How to utilise GPUs in a programmer-friendly way and thereby increase the computational efficiency is the focus of the contribution by Gödekke et al.

Recent advances in processor architectures, based on the introduction of multiple cores, present both possibilities and challenges for developers of CSE software. In their paper, Wallin et al. give an overview of the issues involved.

One key to high performance in numerical comptuations is to represent data in a way that can make efficient use of the computer memory hierarchy and parallel memory units. In software for numerical solution of partial differential equations, the key data structure is the computational mesh. Logg's paper is about a general-purpose C++ mesh class with a storage scheme that allows for both convenient representation of a variety of mesh types and efficient computations with the mesh data.

The final three papers of this special issue address the theme of automatisation. One aspect of this is to provide automatic support for multi-language programming. Multi-language programming is an old idea that has attained renewed interest recently as a consequence of component based software design, where different components can be developed in different languages. For example, it is quite common to develop low-level numerical components in Fortran or C while components for higher-level coordination are implemented in some language more suited to that purpose. In order for these different components to interact smoothly there is need for some kind of automatic support. The paper by Peterson presents one such automatisation effort, for connecting Python and Fortran components.

Automatisation is not least useful for those parts of software development that are highly non-trivial to carry out by hand. Parallelisation is an example. Pflaum and Rahimi describes how to use C++ template techniques to automatise that task in a tricky case: staggered grid-based computations for numerical solution of partial differential equations.

The final contribution of this special issue closes the loop by connecting the theme of automatisation to the topic of software frameworks discussed in the first two papers of the issue. In the design of software frameworks, there is a conflict between efficient execution on one hand and flexibility for the user on the other. For efficient execution, it is preferable that all decisions about specific choices at variation points are made no later than at compile-time, thus allowing for the compiler to make efficient optimisations. On the other hand, the user will have more flexibility if the decisions can be delayed until run-time. In frameworks based on statically compiled languages this represents a challenging conflict. Quintino and DeConinck describe a way to overcome the conflict in a C++-based framework, by using template-based techniques and run-time compilation to delay decisions until run-time, while still producing highly efficient executable code.

The three themes covered by the special issue are strongly interrelated. The main challenge for developers of CSE software is to construct accurate and robust software that simultaneously meets the two conflicting goals of high human efficiency at application development-time and high performance at run-time. We hope that the readers of this special issue will get a good overview of issues involved in this exciting area of research.