PARALLEL SCIENTIFIC COMPUTING AND OPTIMIZATION

Springer Optimization and Its Applications

VOLUME 27

Aims and Scope
Optimization has been expanding in all directions at an astonishing rate during the last few decades. New algorithmic and theoretical techniques have been developed, the diffusion into other disciplines has proceeded at a rapid pace, and our knowledge of all aspects of the field has grown even more profound. At the same time, one of the most striking trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field. Optimization has been a basic tool in all areas of applied mathematics, engineering, medicine, economics and other sciences.

The series *Springer Optimization and Its Applications* publishes undergraduate and graduate textbooks, monographs and state-of-the-art expository works that focus on algorithms for solving optimization problems and also study applications involving such problems. Some of the topics covered include nonlinear optimization (convex and nonconvex), network flow problems, stochastic optimization, optimal control, discrete optimization, multi-objective programming, description of software packages, approximation techniques and heuristic approaches.

PARALLEL SCIENTIFIC COMPUTING AND OPTIMIZATION

Advances and Applications

By

RAIMONDAS ČIEGIS
Vilnius Gediminas Technical University, Lithuania

DAVID HENTY
University of Edinburgh, United Kingdom

BO KÅGSTRÖM
Umeå University, Sweden

JULIUS ŽILINSKAS
Vilnius Gediminas Technical University and Institute of Mathematics and Informatics, Lithuania

Springer

Raimondas Čiegis
Department of Mathematical Modelling
Vilnius Gediminas Technical University
Saulėtekio al. 11
LT-10223 Vilnius
Lithuania
rc@fm.vgtu.lt

David Henty
EPCC
The University of Edinburgh
James Clerk Maxwell Building
Mayfield Road
Edinburgh EH9 3JZ
United Kingdom
d.henty@epcc.ed.ac.uk

Bo Kågström
Department of Computing Science and
High Performance Computing
Center North (HPC2N)
Umeå University
SE-901 87 Umeå
Sweden
bokg@cs.umu.se

Julius Žilinskas
Vilnius Gediminas Technical University and
Institute of Mathematics and Informatics
Akademijos 4
LT-08663 Vilnius
Lithuania
julius.zilinskas@ktl.mii.lt

ISSN: 1931-6828
ISBN: 978-0-387-09706-0 e-ISBN: 978-0-387-09707-7

Library of Congress Control Number: 2008937480

Mathematics Subject Classification (2000): 15-xx, 65-xx, 68Wxx, 90Cxx

Printed on acid-free paper

springer.com

Preface

The book is divided into four parts: Parallel Algorithms for Matrix Computations, Parallel Optimization, Management of Parallel Programming Models and Data, and Parallel Scientific Computing in Industrial Applications.

The first part of the book includes chapters on parallel matrix computations. The chapter by R. Granat, I. Jonsson, and B. Kågström, "RECSY and SCASY Library Software: Recursive Blocked and Parallel Algorithms for Sylvester-type Matrix Equations with Some Applications", gives an overview of state-of-the-art high-performance computing (HPC) algorithms and software for solving various standard and generalized Sylvester-type matrix equations. Computer-aided control system design (CACSD) is a great source of applications for matrix equations including different eigenvalue and subspace problems and for condition estimation. The parallelization is invoked at two levels: *globally* in a distributed memory paradigm, and *locally* on shared memory or multicore nodes as part of the distributed memory environment.

In the chapter by A. Jakušev, R. Čiegis, I. Laukaitytė, and V. Trofimov, "Parallelization of Linear Algebra Algorithms Using ParSol Library of Mathematical Objects", the mathematical objects library ParSol is described and evaluated. It is applied to implement the finite difference scheme to solve numerically a system of PDEs describing a nonlinear interaction of two counter-propagating laser waves.

The chapter by R.L. Muddle, J.W. Boyle, M.D. Mihajlović, and M. Heil, "The Development of an Object-Oriented Parallel Block Preconditioning Framework", is devoted to the analysis of block preconditioners that are applicable to problems that have different types of degree of freedom. The authors discuss the development of an object-oriented parallel block preconditioning framework within OOMPH-LIB, the object-oriented, multi-physics, finite-element library, available as open-source software. The performance of this framework is demonstrated for problems from non-linear elasticity, fluid mechanics, and fluid-structure interaction.

In the chapter by C. Denis, R. Couturier, and F. Jézéquel, "A Sparse Linear System Solver Used in a Distributed and Heterogenous Grid Computing Environment", the GREMLINS (GRid Efficient Linear Systems) solver of systems of linear

equations is developed. The algorithm is based on multisplitting techniques, and a new balancing algorithm is presented.

The chapter by A.G. Sunderland, "Parallel Diagonalization Performance on High-Performance Computers", analyzes the performance of parallel eigensolvers from numerical libraries such as ScaLAPACK on the latest parallel architectures using data sets derived from large-scale scientific applications.

The book continues with the second part focused on parallel optimization. In the chapter by J. Žilinskas, "Parallel Global Optimization in Multidimensional Scaling", global optimization methods are outlined, and global optimization algorithms for multidimensional scaling are reviewed with particular emphasis on parallel computing. Global optimization algorithms are computationally intensive, and solution time crucially depends on the dimensionality of a problem. Parallel computing enables solution of larger problems.

The chapter by K. Woodsend and J. Gondzio, "High-Performance Parallel Support Vector Machine Training", shows how the training process of support vector machines can be reformulated to become suitable for high-performance parallel computing. Data is pre-processed in parallel to generate an approximate low-rank Cholesky decomposition. An optimization solver then exploits the problem's structure to perform many linear algebra operations in parallel, with relatively low data transfer between processors, resulting in excellent parallel efficiency for very-large-scale problems.

The chapter by R. Paulavičius and J. Žilinskas, "Parallel Branch and Bound Algorithm with Combination of Lipschitz Bounds over Multidimensional Simplices for Multicore Computers", presents parallelization of a branch and bound algorithm for global Lipschitz minimization with a combination of extreme (infinite and first) and Euclidean norms over a multidimensional simplex. OpenMP is used to implement the parallel version of the algorithm for multicore computers. The efficiency of the parallel algorithm is studied using an extensive set of multidimensional test functions for global optimization.

The chapter by S. Ivanikovas, E. Filatovas, and J. Žilinskas, "Experimental Investigation of Local Searches for Optimization of Grillage-Type Foundations", presents a multistart approach for optimal pile placement in grillage-type foundations. Various algorithms for local optimization are applied, and their performance is experimentally investigated and compared. Parallel computations is used to speed up experimental investigation.

The third part of the book covers management issues of parallel programs and data. In the chapter by D. Henty and A. Gray, "Comparison of the UK National Supercomputer Services: HPCx and HECToR", an overview of the two current UK national HPC services, HPCx and HECToR, are given. Such results are particularly interesting, as these two machines will now be operating together for some time and users have a choice as to which machine best suits their requirements. Results of extensive experiments are presented.

In the chapter by I.T. Todorov, I.J. Bush, and A.R. Porter, "DL_POLY_3 I/O: Analysis, Alternatives, and Future Strategies", it is noted that an important bottleneck in the scalability and efficiency of any molecular dynamics software is the I/O

speed and reliability as data has to be dumped and stored for postmortem analysis. This becomes increasingly more important when simulations scale to many thousands of processors and system sizes increase to many millions of particles. This study outlines the problems associated with I/O when performing large classic MD runs and shows that it is necessary to use parallel I/O methods when studying large systems.

The chapter by M. Piotrowski, "Mixed Mode Programming on HPCx", presents several benchmark codes based on iterative Jacobi relaxation algorithms: a pure MPI version and three mixed mode (MPI + OpenMP) versions. Their performance is studied and analyzed on mixed architecture – cluster of shared memory nodes. The results show that none of the mixed mode versions managed to outperform the pure MPI, mainly due to longer MPI point-to-point communication times.

The chapter by A. Grothey, J. Hogg, K. Woodsend, M. Colombo, and J. Gondzio, "A Structure Conveying Parallelizable Modeling Language for Mathematical Programming", presents an idea of using a modeling language for the definition of mathematical programming problems with block constructs for description of structure that may make parallel model generation of large problems possible. The proposed structured modeling language is based on the popular modeling language AMPL and implemented as a pre-/postprocessor to AMPL. Solvers based on block linear algebra exploiting interior point methods and decomposition solvers can therefore directly exploit the structure of the problem.

The chapter by R. Smits, M. Kramer, B. Stappers, and A. Faulkner, "Computational Requirements for Pulsar Searches with the Square Kilometer Array", is devoted to the analysis of computational requirements for beam forming and data analysis, assuming the SKA Design Studies' design for the SKA, which consists of 15-meter dishes and an aperture array. It is shown that the maximum data rate from a pulsar survey using the 1-km core becomes about $2.7 \cdot 10^{13}$ bytes per second and requires a computation power of about $2.6 \cdot 10^{17}$ ops for a deep real-time analysis.

The final and largest part of the book covers applications of parallel computing. In the chapter by R. Čiegis, F. Gaspar, and C. Rodrigo, "Parallel Multiblock Multigrid Algorithms for Poroelastic Models", the application of a parallel multigrid method for the two-dimensional poroelastic model is investigated. A domain is partitioned into structured blocks, and this geometrical structure is used to develop a parallel version of the multigrid algorithm. The convergence for different smoothers is investigated, and it is shown that the box alternating line Vanka-type smoother is robust and efficient.

The chapter by V. Starikovičius, R. Čiegis, O. Iliev, and Z. Lakdawala, "A Parallel Solver for the 3D Simulation of Flows Through Oil Filters", presents a parallel solver for the 3D simulation of flows through oil filters. The Navier–Stokes–Brinkmann system of equations is used to describe the coupled laminar flow of incompressible isothermal oil through open cavities and cavities with filtering porous media. Two parallel algorithms are developed on the basis of the sequential numerical algorithm. The performance of implementations of both algorithms is studied on clusters of multicore computers.

The chapter by S. Eastwood, P. Tucker, and H. Xia, "High-Performance Computing in Jet Aerodynamics", is devoted to the analysis of methods for reduction of the noise generated by the propulsive jet of an aircraft engine. The use of high-performance computing facilities is essential, allowing detailed flow studies to be carried out that help to disentangle the effects of numerics from flow physics. The scalability and efficiency of the presented parallel algorithms are investigated.

The chapter by G. Jankevičiūtė and R. Čiegis, "Parallel Numerical Solver for the Simulation of the Heat Conduction in Electrical Cables", is devoted to the modeling of the heat conduction in electrical cables. Efficient parallel numerical algorithms are developed to simulate the heat transfer in cable bundles. They are implemented using MPI and targeted for distributed memory computers, including clusters of PCs.

The chapter by A. Deveikis, "Orthogonalization Procedure for Antisymmetrization of J-shell States", presents an efficient procedure for construction of the antisymmetric basis of j-shell states with isospin. The approach is based on an efficient algorithm for construction of the idempotent matrix eigenvectors, and it reduces to an orthogonalization procedure. The presented algorithm is much faster than the diagonalization routine `rs()` from the EISPACK library.

In the chapter by G.A. Siamas, X. Jiang, and L.C. Wrobel, "Parallel Direct Numerical Simulation of an Annular Gas–Liquid Two-Phase Jet with Swirl", the flow characteristics of an annular swirling liquid jet in a gas medium is examined by direct solution of the compressible Navier–Stokes equations. A mathematical formulation is developed that is capable of representing the two-phase flow system while the volume of fluid method has been adapted to account for the gas compressibility. Fully 3D parallel direct numerical simulation (DNS) is performed utilizing 512 processors, and parallelization of the code was based on domain decomposition.

In the chapter by I. Laukaitytė, R. Čiegis, M. Lichtner, and M. Radziunas, "Parallel Numerical Algorithm for the Traveling Wave Model", a parallel algorithm for the simulation of the dynamics of high-power semiconductor lasers is presented. The model equations describing the multisection broad-area semiconductor lasers are solved by the finite difference scheme constructed on staggered grids. The algorithm is implemented by using the ParSol tool of parallel linear algebra objects.

The chapter by X. Guo, M. Pinna, and A.V. Zvelindovsky, "Parallel Algorithm for Cell Dynamics Simulation of Soft Nano-Structured Matter", presents a parallel algorithm for large-scale cell dynamics simulation. With the efficient strategy of domain decomposition and the fast method of neighboring points location, simulations of large-scale systems have been successfully performed.

The chapter by Ž. Dapkūnas and J. Kulys, "Docking and Molecular Dynamics Simulation of Complexes of High and Low Reactive Substrates with Peroxidases", presents docking and parallel molecular dynamics simulations of two peroxidases (*ARP* and *HRP*) and two compounds (*LUM* and *IMP*). The study of docking simulations gives a clue to the reason of different reactivity of *LUM* and similar reactivity of *IMP* toward two peroxidases. In the case of *IMP*, $-$OH group is near FE=O in both peroxidases, and hydrogen bond formation between $-$OH and Fe=0 is

possible. In the case of *LUM*, N-H is near Fe=0 in *ARP* and the hydrogen formation is possible, but it is farther in *HRP* and hydrogen bond with Fe=O is not formed.

The works were presented at the bilateral workshop of British and Lithuanian scientists "High Performance Scientific Computing" held in Druskininkai, Lithuania, 5–8 February 2008. The workshop was supported by the British Council through the INYS program.

The British Council's International Networking for Young Scientists program (INYS) brings together young researchers from the UK and other countries to make new contacts and promote the creative exchange of ideas through short conferences. Mobility for young researchers facilitates the extended laboratory in which all researchers now operate: it is a powerful source of new ideas and a strong force for creativity. Through the INYS program, the British Council helps to develop high-quality collaborations in science and technology between the UK and other countries and shows the UK as a leading partner for achievement in world science, now and in the future. The INYS program is unique in that it brings together scientists in any priority research area and helps develop working relationships. It aims to encourage young researchers to be mobile and expand their knowledge. The INYS supported workshop "High Performance Scientific Computing" was organized by the University of Edinburgh, UK, and Vilnius Gediminas Technical University, Lithuania. The meeting was coordinated by Professor R. Čiegis and Dr. J. Žilinskas from Vilnius Gediminas Technical University and Dr. D. Henty from The University of Edinburgh. The homepage of the workshop is available at http://techmat.vgtu.lt/~inga/inys2008/.

Twenty-four talks were selected from thirty-two submissions from young UK and Lithuanian researchers. Professor B. Kågström from Umeå University, Sweden, and Dr. I. Todorov from Daresbury Laboratory, UK, gave invited lectures. Review lectures were also given by the coordinators of the workshop.

This book contains review papers and revised contributed papers presented at the workshop. All twenty-three papers have been reviewed. We are very thankful to the reviewers for their recommendations and comments.

We hope that this book will serve as a valuable reference document for the scientific community and will contribute to the future cooperation between the participants of the workshop.

We would like to thank the British Council for financial support. We are very grateful to the managing editor of the series, Professor Panos Pardalos, for his encouragement.

Druskininkai, Lithuania
February 2008

Raimondas Čiegis
David Henty
Bo Kågström
Julius Žilinskas

Contents

List of Contributors

Jonathan W. Boyle
School of Mathematics, University of Manchester, Oxford Road, Manchester, M13 9PL, UK, e-mail: j.boyle@manchester.ac.uk

Ian J. Bush
STFC Daresbury Laboratory, Warrington WA4 4AD, UK

Marco Colombo
School of Mathematics, University of Edinburgh, Edinburgh, UK

Raphael Couturier
Laboratoire d Informatique del Université de Franche-Comté, BP 527, 90016 Belfort Cedex, France, e-mail: Raphael.Couturier@iut-bm.univ-fcomte.fr

Raimondas Čiegis
Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223 Vilnius, Lithuania, e-mail: rc@fm.vgtu.lt

Žilvinas Dapkūnas
Vilnius Gediminas Technical University, Department of Chemistry and Bioengineering, Saulėtekio Avenue 11, LT-10223 Vilnius, Lithuania, e-mail: Zilvinas.Dapkunas@fm.vgtu.lt

Christophe Denis
School of Electronics, Electrical Engineering & Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, UK, e-mail: C.Denis@qub.ac.uk
UPMC Univ Paris 06, Laboratoire d'Informatique LIP6, 4 place Jussieu, 75252 Paris Cedex 05, France, e-mail: Christophe.Denis@lip6.fr

Algirdas Deveikis
Department of Applied Informatics, Vytautas Magnus University, Kaunas, Lithuania, e-mail: a.deveikis@if.vdu.lt

Simon Eastwood
Whittle Laboratory, University of Cambridge, Cambridge, UK,
e-mail: se282@cam.ac.uk

Andrew Faulkner
Jodrell Bank Centre for Astrophysics, University of Manchester, Manchester, UK

Ernestas Filatovas
Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania, e-mail: ernest.filatov@gmail.com

Francisco Gaspar
Departamento de Matematica Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain, e-mail: fjgaspar@unizar.es

Jacek Gondzio
School of Mathematics, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JZ, UK, e-mail: j.gondzio@ed.ac.uk

Robert Granat
Department of Computing Science and HPC2N, Umeå University, Umeå, Sweden, e-mail: granat@cs.umu.se

Alan Gray
Edinburgh Parallel Computing Centre, The University of Edinburgh, Edinburgh, UK, e-mail: a.gray@epcc.ed.ac.uk

Andreas Grothey
School of Mathematics, University of Edinburgh, Edinburgh, UK,
e-mail: A.Grothey@ed.ac.uk

Xiaohu Guo
School of Computing, Engineering and Physical Sciences, University of Central Lancashire, Preston, Lancashire, PR1 2HE, UK, e-mail: XGuo@uclan.ac.uk

Matthias Heil
School of Mathematics, University of Manchester, Oxford Road, Manchester, M13 9PL, UK, e-mail: m.heil@maths.man.ac.uk

David Henty
Edinburgh Parallel Computing Centre, The University of Edinburgh, Edinburgh, UK, e-mail: d.henty@epcc.ed.ac.uk

Jonathan Hogg
School of Mathematics, University of Edinburgh, Edinburgh, UK

Oleg Iliev
Fraunhofer ITWM, Fraunhofer-Platz 1, D-67663 Kaiserslautern, Germany,
e-mail: iliev@itwm.fhg.de

Sergėjus Ivanikovas
Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania, e-mail: ivanikovas@gmail.com

Alexander Jakušev
Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223, Vilnius, Lithuania, e-mail: alexj@fm.vgtu.lt

Gerda Jankevičiūtė
Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223, Vilnius, Lithuania, e-mail: Gerda.Jankeviciute@fm.vgtu.lt

Fabienne Jézéquel
UPMC Univ Paris 06, Laboratoire d'Informatique LIP6, 4 place Jussieu, 75252 Paris Cedex 05, France, e-mail: Fabienne.Jezequel@lip6.fr

Xi Jiang
Brunel University, Mechanical Engineering, School of Engineering and Design, Uxbridge, UB8 3PH, UK, e-mail: Xi.Jiang@brunel.ac.uk

Isak Jonsson
Department of Computing Science and HPC2N, Umeå University, Umeå, Sweden, e-mail: isak@cs.umu.se

Bo Kågström
Department of Computing Science and HPC2N, Umeå University, Umeå, Sweden, e-mail: bokg@cs.umu.se

Michael Kramer
Jodrell Bank Centre for Astrophysics, University of Manchester, Manchester, UK

Juozas Kulys
Vilnius Gediminas Technical University, Department of Chemistry and Bioengineering, Saulėtekio Avenue 11, LT-10223 Vilnius, Lithuania, e-mail: Juozas.Kulys@fm.vgtu.lt

Zhara Lakdawala
Fraunhofer ITWM, Fraunhofer-Platz 1, D-67663 Kaiserslautern, Germany, e-mail: lakdawala@itwm.fhg.de

Inga Laukaitytė
Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223, Vilnius, Lithuania, e-mail: Inga.Laukaityte@fm.vgtu.lt

Mark Lichtner
Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstarsse 39, 10117 Berlin, Germany, e-mail: lichtner@wias-berlin.de

Milan D. Mihajlović
School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK, e-mail: milan@cs.man.ac.uk

Richard L. Muddle
School of Computer Science, University of Manchester, Oxford Road, Manchester, M13 9PL, UK, e-mail: rlm@cs.man.ac.uk

Remigijus Paulavičius
Vilnius Pedagogical University, Studentu 39, LT-08106 Vilnius, Lithuania,
e-mail: r.paulavicius@vpu.lt
Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius,
Lithuania

Marco Pinna
School of Computing, Engineering and Physical Sciences, University of Central
Lancashire, Preston, Lancashire, PR1 2HE, UK, e-mail: MPinna@uclan.ac.uk

Michał Piotrowski
Edinburgh Parallel Computing Centre, University of Edinburgh, Edinburgh, UK,
e-mail: mpiotrow@epcc.ed.ac.uk

Andrew R. Porter
STFC Daresbury Laboratory, Warrington WA4 4AD, UK

Mindaugas Radziunas
Weierstrass Institute for Applied Analysis and Stochastics, Mohrenstarsse 39,
10117 Berlin, Germany, e-mail: radziunas@wias-berlin.de

Carmen Rodrigo
Departamento de Matematica Aplicada, Universidad de Zaragoza, 50009 Zaragoza,
Spain, e-mail: carmen.rodrigo.cardiel@gmail.com

George A. Siamas
Brunel University, Mechanical Engineering, School of Engineering and Design,
Uxbridge, UB8 3PH, UK, e-mail: George.Siamas@brunel.ac.uk

Roy Smits
Jodrell Bank Centre for Astrophysics, University of Manchester, Manchester, UK,
e-mail: Roy.Smits@manchester.ac.uk

Ben Stappers
Jodrell Bank Centre for Astrophysics, University of Manchester, Manchester, UK

Vadimas Starikovičius
Vilnius Gediminas Technical University, Saulėtekio al. 11, LT-10223 Vilnius,
Lithuania, e-mail: vs@sc.vgtu.lt

Andrew G. Sunderland
STFC Daresbury Laboratory, Warrington, UK, e-mail: a.g.sunderland@dl.ac.uk

Ilian T. Todorov
STFC Daresbury Laboratory, Warrington WA4 4AD, UK,
e-mail: i.t.todorov@dl.ac.uk

Vyacheslav Trofimov
M. V. Lomonosov Moscow State University, Vorob'evy gory, 119992, Russia,
e-mail: vatro@cs.msu.su

Paul Tucker
Whittle Laboratory, University of Cambridge, Cambridge, UK

Kristian Woodsend
School of Mathematics, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JZ, UK, e-mail: k.j.woodsend@sms.ed.ac.uk

Luiz C. Wrobel
Brunel University, Mechanical Engineering, School of Engineering and Design, Uxbridge, UB8 3PH, UK, e-mail: Luiz.Wrobel@brunel.ac.uk

Hao Xia
Whittle Laboratory, University of Cambridge, Cambridge, UK

Andrei V. Zvelindovsky
School of Computing, Engineering and Physical Sciences, University of Central Lancashire, Preston, Lancashire, PR1 2HE, UK, e-mail: AVZvelindovsky@uclan.ac.uk

Julius Žilinskas
Institute of Mathematics and Informatics, Akademijos 4, LT-08663 Vilnius, Lithuania, e-mail: julius.zilinskas@ktl.mii.lt
Vilnius Gediminas Technical University, Saulėtekio 11, LT-10223 Vilnius, Lithuania, e-mail: zilinskasjulius@gmail.com

Part I
Parallel Algorithms for Matrix Computations

RECSY and SCASY Library Software: Recursive Blocked and Parallel Algorithms for Sylvester-Type Matrix Equations with Some Applications

Robert Granat, Isak Jonsson, and Bo Kågström

Abstract In this contribution, we review state-of-the-art high-performance computing software for solving common standard and generalized continuous-time and discrete-time Sylvester-type matrix equations. The analysis is based on RECSY and SCASY software libraries. Our algorithms and software rely on the standard Schur method. Two ways of introducing blocking for solving matrix equations in reduced (quasi-triangular) form are reviewed. Most common is to perform a fix block partitioning of the matrices involved and rearrange the loop nests of a single-element algorithm so that the computations are performed on submatrices (matrix blocks). Another successful approach is to combine recursion and blocking.

We consider parallelization of algorithms for reduced matrix equations at two levels: globally in a distributed memory paradigm, and locally on shared memory or multicore nodes as part of the distributed memory environment. Distributed wavefront algorithms are considered to compute the solution to the reduced triangular systems. Parallelization of recursive blocked algorithms is done in two ways. The simplest way is so-called implicit data parallelization, which is obtained by using SMP-aware implementations of level 3 BLAS. Complementary to this, there is also the possibility of invoking task parallelism. This is done by explicit parallelization of independent tasks in a recursion tree using OpenMP. A brief account of some software issues for the RECSY and SCASY libraries is given. Theoretical results are confirmed by experimental results.

1 Motivation and Background

Matrix computations are fundamental and ubiquitous in computational science and its vast application areas. Along with the computer evolution, there is a continuing

Robert Granat · Isak Jonsson · Bo Kågström
Department of Computing Science and HPC2N, Umeå University, Sweden
e-mail: {granat · isak · bokg}@cs.umu.se

demand for new and improved algorithms and library software that is portable, robust, and efficient [13]. In this contribution, we review state-of-the-art high-performance computing (HPC) software for solving common standard and generalized continuous-time (CT) and discrete-time (DT) Sylvester-type matrix equations, see Table 1. Computer-aided control system design (CACSD) is a great source of applications for matrix equations including different eigenvalue and subspace problems and for condition estimation.

Both the RECSY and SCASY software libraries distinguish between *one-sided* and *two-sided* matrix equations. For one-sided matrix equations, the solution is only involved in matrix products of two matrices, e.g., $\mathrm{op}(A)X$ or $X\mathrm{op}(A)$, where $\mathrm{op}(A)$ can be A or A^T. In two-sided matrix equations, the solution is involved in matrix products of three matrices, both to the left and to the right, e.g., $\mathrm{op}(A)X\mathrm{op}(B)$. The more complicated data dependency of the two-sided equations is normally addressed in blocked methods from complexity reasons.

Solvability conditions for the matrix equations in Table 1 are formulated in terms of non-intersecting spectra of standard or generalized eigenvalues of the involved coefficient matrices and matrix pairs, respectively, or equivalently by nonzero associated sep-functions (see Sect. 5 and, e.g., [28, 24, 30] and the references therein).

The rest of this chapter is structured as follows. First, in Sect. 2, variants of the standard Schur method for solving Sylvester-type matrix equations are briefly described. In Sect. 3, two blocking strategies for dense linear algebra computations are discussed and applied to matrix equations in reduced (quasi-triangular) form. Section 4 reviews parallel algorithms for reduced matrix equations based on the explicitly blocked and the recursively blocked algorithms discussed in the previous section. Condition estimation of matrix equations and related topics are discussed in Sect. 5. In Sect. 6, a brief account of some software issues for the RECSY and SCASY libraries are given. Section 7 presents some performance results with focus on the hybrid parallelization model including message passing and multithreading. Finally, Sect. 8 is devoted to some CACSD applications, namely condition estimation of invariant subspaces of Hamiltonian matrices and periodic matrix equations.

Table 1 Considered standard and generalized matrix equations. CT and DT denote the *continuous-time* and *discrete-time* variants, respectively.

Name	Matrix equation	Acronym
Standard CT Sylvester	$AX - XB = C \in \mathbb{R}^{m\times n}$	SYCT
Standard CT Lyapunov	$AX + XA^T = C \in \mathbb{R}^{m\times m}$	LYCT
Generalized Coupled Sylvester	$(AX - YB, DX - YE) = (C,F) \in \mathbb{R}^{(m\times n)\times 2}$	GCSY
Standard DT Sylvester	$AXB - X = C \in \mathbb{R}^{m\times n}$	SYDT
Standard DT Lyapunov	$AXA^T - X = C \in \mathbb{R}^{m\times m}$	LYDT
Generalized Sylvester	$AXB^T - CXD^T = E \in \mathbb{R}^{m\times n}$	GSYL
Generalized CT Lyapunov	$AXE^T + EXA^T = C \in \mathbb{R}^{m\times m}$	GLYCT
Generalized DT Lyapunov	$AXA^T - EXE^T = C \in \mathbb{R}^{m\times m}$	GLYDT

2 Variants of Bartels–Stewart's Schur Method

Our algorithms and software rely on the standard Schur method proposed already in 1972 by Bartels and Stewart [6]. The generic standard method consists of four major steps: (1) initial transformation of the left-hand coefficient matrices to Schur form; (2) updates of the right-hand-side matrix with respect to the orthogonal transformation matrices from the first step; (3) computation of the solution of the reduced matrix equation from the first two steps in a combined forward and backward substitution process; (4) a retransformation of the solution from the third step, in terms of the orthogonal transformations from the first step, to get the solution of the original matrix equation.

Let us demonstrate the solution process by considering the SYCT equation $AX - XB = C$. Step 1 produces the Schur factorizations $T_A = Q^T AQ$ and $T_B = P^T BP$, where $Q \in \mathbb{R}^{m\times m}$ and $P \in \mathbb{R}^{n\times n}$ are orthogonal and T_A and T_B are upper quasi-triangular, i.e., having 1×1 and 2×2 diagonal blocks corresponding to real and complex conjugate pairs of eigenvalues, respectively. Reliable and efficient algorithms for the Schur reduction step can be found in LAPACK [4] and in ScaLAPACK [26, 7] for distributed memory (DM) environments.

Steps 2 and 4 are typically conducted by two consecutive matrix multiply (GEMM) operations [12, 33, 34]: $\tilde{C} = Q^T CP$ and $X = Q\tilde{X}P^T$, where $\tilde{X}$ is the solution to the triangular equation

$$T_A\tilde{X} - \tilde{X}T_B = \tilde{C},$$

resulting from steps 1 and 2, and solved in step 3. Similar Schur-based methods are formulated for the standard equations LYCT, LYDT, and SYDT.

It is also straightforward to extend the generic Bartels–Stewart method to the generalized matrix equations GCSY, GSYL, GLYCT, and GLYDT. Now, we must rely on robust and efficient algorithms and software for the generalized Schur reduction (Hessenberg-triangular reduction and the QZ algorithm) [40, 39, 10, 1, 2, 3, 32], and algorithms for solving triangular forms of generalized matrix equations.

For illustration we consider GCSY, the generalized coupled Sylvester equation $(AX - YB, DX - YE) = (C, F)$ [37, 36]. In step 1, (A, D) and (B, E) are transformed to generalized real Schur form, i.e., $(T_A, T_D) = (Q_1^T AZ_1, Q_1^T DZ_1)$, $(T_B, T_E) = (Q_2^T BZ_2, Q_2^T EZ_2)$, where T_A and T_B are upper quasi-triangular, T_D and T_E are upper triangular, and $Q_1, Z_1 \in \mathbb{R}^{m\times m}$, $Q_2, Z_2 \in \mathbb{R}^{n\times n}$ are orthogonal. Step 2 updates the right-hand-side matrices $(\tilde{C}, \tilde{F}) = (Q_1^T CZ_2, Q_1^T FZ_2)$ leading to the reduced triangular GCSY

$$(T_A\tilde{X} - \tilde{Y}T_B, T_D\tilde{X} - \tilde{Y}T_E) = (\tilde{C}, \tilde{F}),$$

which is solved in step 3. The solution (X, Y) of the original GCSY is obtained in step 4 by the orthogonal equivalence transformation $(X, Y) = (Z_1\tilde{X}Z_2^T, Q_1\tilde{Y}Q_2^T)$.

Notice that care has to be taken to preserve the symmetry of the right-hand-side C in the Lyapunov equations LYCT, LYDT, GLYCT, and GLYDT (see Table 1) during each step of the different variants of the Bartels–Stewart method. This is important

for reducing the complexity in steps 2–4 (for step 3, see Sect. 3) and to ensure that the computed solution X of the corresponding Lyapunov equation is guaranteed to be symmetric on output.

In general, the computed solutions of the matrix equations in Table 1 overwrite the corresponding right-hand-side matrices of the respective equation.

3 Blocking Strategies for Reduced Matrix Equations

In this section, we review two ways of introducing blocking for solving matrix equations in reduced (quasi-triangular) form. Most common is to perform a fix block partitioning of the matrices involved and rearrange the loop nests of a single-element algorithm so that the computations are performed on submatrices (matrix blocks). This means that the operations in the inner-most loops are expressed in matrix-matrix operations that can deliver high performance via calls to optimized level 3 BLAS. Indeed, this *explicit blocking* approach is extensively used in LAPACK [4].

Another successful approach is to combine recursion and blocking, leading to an automatic variable blocking with the potential for matching the deep memory hierarchies of today's HPC systems. *Recursive blocking* means that the involved matrices are split in the middle (by columns, by rows, or both) and a number of smaller problems are generated. In turn, the subarrays involved are split again generating a number of even smaller problems. The divide phase proceeds until the size of the controlling recursion blocking parameter becomes smaller than some predefined value. This termination criteria ensures that all leaf operations in the recursion tree are substantial level 3 (matrix-matrix) computations. The conquer phase mainly consists of increasingly sized matrix multiply and add (GEMM) operations. For an overview of recursive blocked algorithms and hybrid data structures for dense matrix computations and library software, we refer to the SIAM Review paper [13].

In the next two subsections, we demonstrate how explicit and recursive blocking are applied to solving matrix equations already in reduced form. To solve each of the Sylvester equations SYCT, SYDT, GCSY, and GSYL in quasi-triangular form is an $O(m^2n+mn^2)$ operation. Likewise, solving reduced Lyapunov equations LYCT, LYDT, GLYCT, and GLYDT are all $O(m^3)$ operations. The blocked methods described below have a similar complexity, assuming that the more complicated data dependencies of the two-sided matrix equations (SYDT, LYDT, GSYL, GLYCT, GLYDT) are handled appropriately.

3.1 Explicit Blocked Methods for Reduced Matrix Equations

We apply explicit blocking to reformulate each matrix equation problem into as much level 3 BLAS operations as possible. In the following, the (i, j)th block of a

partitioned matrix, say X, is denoted X_{ij}. For illustration of the basic concepts, we consider the one-sided matrix equation SYCT and the two-sided LYDT. Let m_b and n_b be the block sizes used in an explicit block partitioning of the matrices A and B, respectively. In turn, this imposes a similar block partitioning of C and X (which overwrites C). Then $D_l = \lceil m/m_b \rceil$ and $D_r = \lceil n/n_b \rceil$ are the number of diagonal blocks in A and B, respectively. Now, SYCT can be rewritten in block-partitioned form as

$$A_{ii}X_{ij} - X_{ij}B_{jj} = C_{ij} - (\sum_{k=i+1}^{D_l} A_{ik}X_{kj} - \sum_{k=1}^{j-1} X_{ik}B_{kj}), \tag{1}$$

for $i = 1,2,\ldots,D_l$ and $j = 1,2,\ldots,D_r$. This block formulation of SYCT can be implemented using a couple of nested loops that call a node (or kernel) solver for the $D_l \cdot D_r$ small matrix equations and level 3 operations in the right-hand side [22]. Similarly, we express LYDT in explicit blocked form as

$$A_{ii}X_{ij}A_{jj}^T - X_{ij} = C_{ij} - \sum_{(k,l)=(i,j)}^{(D_l,D_l)} A_{ik}X_{kl}A_{jl}^T,\ (k,l) \neq (i,j). \tag{2}$$

Notice that the blocking of LYDT decomposes the problem into several smaller SYDT ($i \neq j$) and LYDT ($i = j$) equations. Apart from solving for only the upper or lower triangular part of X in the case of LYDT, a complexity of $O(m^3)$ can be retained by using a technique that stores *intermediate sums of matrix products* to avoid computing certain matrix products in the right-hand side of (2) several times. The same technique can also be applied to all two-sided matrix equations.

Similarly, explicit blocking and the complexity reduction technique are applied to the generalized matrix equations. Here, we illustrate with the one-sided GCSY, and the explicit blocked variant takes the form

$$\begin{cases} A_{ii}X_{ij} - Y_{ij}B_{jj} = C_{ij} - (\Sigma_{k=i+1}^{D_l} A_{ik}X_{kj} - \Sigma_{k=1}^{j-1} Y_{ik}B_{kj}) \\ D_{ii}X_{ij} - Y_{ij}E_{jj} = F_{ij} - (\Sigma_{k=i+1}^{D_l} D_{ik}X_{kj} - \Sigma_{k=1}^{j-1} X_{ik}E_{kj}), \end{cases} \tag{3}$$

where $i = 1,2,\ldots,D_l$ and $j = 1,2,\ldots,D_r$. A resulting serial level 3 algorithm is implemented as a couple of nested loops over the matrix operations defined by (3).

We remark that all linear matrix equations considered can be rewritten as an equivalent large linear system of equations $Zx = y$, where Z is the Kronecker product representation of the corresponding Sylvester-type operator. This is utilized in condition estimation algorithms and kernel solvers for small-sized matrix equations (see Sects. 5 and 6.1).

3.2 Recursive Blocked Methods for Reduced Matrix Equations

To make it easy to compare the two blocking strategies, we start by illustrating recursive blocking for SYCT.

The sizes of m and n control three alternatives for doing a *recursive splitting*. In Case 1 ($1 \leq n \leq m/2$), A is split by rows and columns, and C by rows only. In Case 2 ($1 \leq m \leq n/2$), B is split by rows and columns, and C by columns only. Finally, in Case 3 ($n/2 < m < 2n$), both rows and columns of the matrices A, B, and C are split:

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} - \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}.$$

This recursive splitting results in the following four triangular SYCT equations:

$$\begin{aligned} A_{11}X_{11} - X_{11}B_{11} &= C_{11} - A_{12}X_{21}, \\ A_{11}X_{12} - X_{12}B_{22} &= C_{12} - A_{12}X_{22} + X_{11}B_{12}, \\ A_{22}X_{21} - X_{21}B_{11} &= C_{21}, \\ A_{22}X_{22} - X_{22}B_{22} &= C_{22} + X_{21}B_{12}. \end{aligned}$$

Conceptually, we start by solving for X_{21} in the third equation above. After updating C_{11} and C_{22} with respect to X_{21}, one can solve for X_{11} and X_{22}. Both updates and the triangular Sylvester solves are independent operations and can be executed concurrently. Finally, C_{12} is updated with respect to X_{11} and X_{22}, and we can solve for X_{12}. The description above defines a recursion template that is applied to the four Sylvester sub-solves leading to a recursive blocked algorithm that terminates with calls to optimized kernel solvers for the leaf computations of the recursion tree.

We also illustrate the recursive blocking and template for GSYL, the most general of the two-sided matrix equations. Also here, we demonstrate Case 3 (Cases 1 and 2 can be seen as special cases), where (A,C), (B,D), and E are split by rows and columns:

$$\begin{aligned} &\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} B_{11}^T & \\ B_{12}^T & B_{22}^T \end{bmatrix} \\ &- \begin{bmatrix} C_{11} & C_{12} \\ & C_{22} \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \begin{bmatrix} D_{11}^T & \\ D_{12}^T & D_{22}^T \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}, \end{aligned}$$

leading to the following four triangular GSYL equations:

$$\begin{aligned} A_{11}X_{11}B_{11}^T - C_{11}X_{11}D_{11}^T &= E_{11} - A_{12}X_{21}B_{11}^T - (A_{11}X_{12} + A_{12}X_{22})B_{12}^T \\ &\quad + C_{12}X_{21}D_{11}^T + (C_{11}X_{12} + C_{12}X_{22})D_{12}^T, \\ A_{11}X_{12}B_{22}^T - C_{11}X_{12}D_{22}^T &= E_{12} - A_{12}X_{22}B_{22}^T + C_{12}X_{22}D_{22}^T, \\ A_{22}X_{21}B_{11}^T - C_{22}X_{21}D_{11}^T &= E_{21} - A_{22}X_{22}B_{12}^T + C_{22}X_{22}D_{12}^T, \\ A_{22}X_{22}B_{22}^T - C_{22}X_{22}D_{22}^T &= E_{22}. \end{aligned}$$

Now, we start by solving for X_{22} in the fourth equation above. After updating E_{12} and E_{21} with respect to X_{22}, we can solve for X_{12} and X_{21}. As for SYCT, both updates and the triangular GSYL solves are independent operations and can be executed concurrently. Finally, after updating E_{11} with respect to X_{12}, X_{21} and X_{22}, we solve for X_{11}. Some of the updates of E_{11} can be combined in larger GEMM operations,