# CONFERENTIE
# VAN NUMERIEK WISKUNDIGEN

26 - 28 september 1994

CONFERENTIEOORD WOUDSCHOTEN

ZEIST

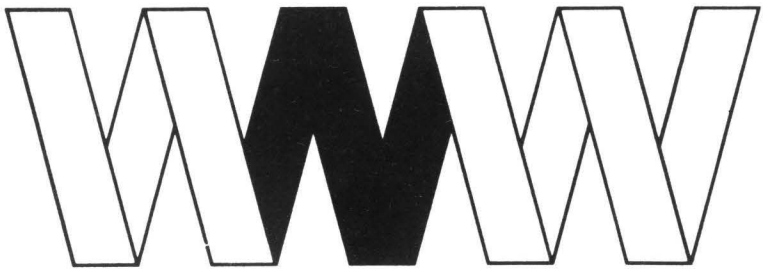
Werkgemeenschap Numerieke Wiskunde

# CONFERENTIE
# VAN NUMERIEK WISKUNDIGEN

26 - 28 september 1994

CONFERENTIEOORD WOUDSCHOTEN

ZEIST



Werkgemeenschap Numerieke Wiskunde

# NEGENTIENDE CONFERENTIE NUMERIEKE WISKUNDE

## Doel van de conferentie

De Conferentie Numerieke Wiskunde wordt eenmaal per jaar gehouden onder auspiciën van de Werkgemeenschap Numerieke Wiskunde. Het doel van de conferentie is kennis te nemen van recente ontwikkelingen binnen de numerieke wiskunde. Hiertoe worden jaarlijks twee thema's vastgesteld. Enkele (buitenlandse) deskundigen worden uitgenodigd over deze thema's lezingen te houden.

## Thema's

**A.** Diverse aspecten van High Performance Computing and Networking

**B.** Differentiaal-Algebraïsche vergelijkingen

## Organisatie

De organisatie is in handen van de voorbereidingscommissie bestaande uit R.M.M. Mattheij (TUE)(voorzitter), J. de Groot (Philips), H.A. van der Vorst (RUU), M. van Veldhuizen (VU) en B.P. Sommeijer (CWI)(secretaris). Organisatorische medewerking is verleend door het Centrum voor Wiskunde en Informatica. Financiële ondersteuning is gegeven door de Wetenschappelijke Raad van de Stichting Mathematisch Centrum. Tevens is er een financiële bijdrage geschonken door Computing & Systems Consultants bv (Eindhoven).

Nineteenth Dutch Conference on Numerical Analysis

**Themes and Speakers**

Theme A. *Various aspects of High Performance Computing and Networking*

W. Joppich (GMD, Germany)
D. E. Keyes (ICASE, USA)
S. Vandewalle (Caltech, USA)
H.A.G. Wijshoff (Leiden University, The Netherlands)

Theme B. *Differential-Algebraic equations*

C. Führer (DLR, Germany)
E. Hairer (University of Geneva, Switzerland)

Short presentations will be given by:

E. Brakkee, Delft University of Technology (Theme A)
M. Neytcheva, Nijmegen University (Theme A)
J.J.B. de Swart, CWI Amsterdam (Theme A)
K.H. Tan/M.J.A. Borsboom, Utrecht University/Delft Hydraulics (Theme A)
M.J.C. van de Wiel, Philips Research Laboratories (Theme B)

**Program and titles of lectures**

*Monday, September 26*

| | | |
|---|---|---|
| 10.00 - 11.10 | **arrival, coffee** | |
| 11.10 | **opening** | |
| 11.15 | **D.E. Keyes** | |
| | 11.15 - 12.05: | Domain Decomposition: A Bridge between Nature and Parallel Computers |
| | 12.05 - 12.15: | discussion |
| 12.20 | **lunch** | |
| 13.45 | **E. Hairer** | |
| | 13.45 - 14.35: | Runge-Kutta-Type Methods for Differential-Algebraic Systems |
| | 14.35 - 14.45: | discussion |
| 14.50 | **S. Vandewalle** | |
| | 14.50 - 15.40: | Recent Developments in Waveform Relaxation Methods |
| | 15.40 - 15.50: | discussion |
| 15.50 | **tea** | |
| 16.15 | **C. Führer** | |
| | 16.15 - 17.05: | Multistep Methods for Differential-Algebraic Equations in Euler-Lagrange Form |
| | 17.05 - 17.15: | discussion |
| 17.20 | **E. Brakkee** | |
| | 17.20 - 17.45: | Parallel Solution of the Incompressible Navier-Stokes Equations |
| | 17.45 - 17.50: | discussion |
| 18.15 | **dinner** | |

*Tuesday, September 27*

| | |
|---|---|
| 08.00 | **breakfast** |
| 09.00 | **W. Joppich**<br>09.00 - 09.50: High Performance Computing and Networking, Applications, Part I<br>09.50 - 10.00: discussion |
| 10.05 | **M.J.C. van de Wiel**<br>10.05 - 10.30: Differential-Algebraic Equations in Circuit Analysis<br>10.30 - 10.35: discussion |
| 10.35 | **coffee** |
| 11.10 | **H.A.G. Wijshoff**<br>11.10 - 12.00: Automatic Data Structure Selection and Transformation for Sparse Matrix Computations, Part I<br>12.00 - 12.10: discussion |
| 12.15 | **J.J.B. de Swart**<br>12.15 - 12.40: Diagonally Implicit Iteration Methods for IVP Solvers<br>12.40 - 12.45: discussion |
| 12.50 | **lunch** |
| 14.15 | **D.E. Keyes**<br>14.15 - 15.05: Newton-Krylov-Schwarz Domain Decomposition Methods for Partial Differential Equations in Science and Engineering<br>15.05 - 15.15: discussion |
| 15.20 | **K.H. Tan/M.J.A. Borsboom**<br>15.20 - 15.45: Optimized Parallel Block Preconditioners Based on Domain Decomposition<br>15.45 - 15.50: discussion |
| 15.50 | **tea** |
| 16.15 | **E. Hairer**<br>16.15 - 17.05: Long-Time Integration of Constrained Hamiltonian Systems<br>17.05 - 17.15: discussion |
| 17.20 | **Meeting of the Dutch "Werkgemeenschap Numerieke Wiskunde"** |
| 18.00 | **dinner** |

*Wednesday, September 28*

| | | |
|---|---|---|
| 08.00 | **breakfast** | |
| 09.00 | **S. Vandewalle** | |
| | 09.00 - 09.50: | Space-time Iterative Methods for Time-Dependent Partial Differential Equations |
| | 09.50 - 10.00: | discussion |
| 10.05 | **M. Neytcheva** | |
| | 10.05 - 10.30: | On the scalability of the Algebraic Multilevel Iteration Method |
| | 10.30 - 10.35: | discussion |
| 10.35 | **coffee** | |
| 11.10 | **C. Führer** | |
| | 11.10 - 12.00: | β-blocked Multistep Methods for Euler-Lagrange DAEs |
| | 12.00 - 12.10: | discussion |
| 12.15 | **lunch** | |
| 13.30 | **W. Joppich** | |
| | 13.30 - 14.20: | High Performance Computing and Networking, Applications, Part II |
| | 14.20 - 14.30: | discussion |
| 14.35 | **H.A.G. Wijshoff** | |
| | 14.35 - 15.25: | Automatic Data Structure Selection and Transformation for Sparse Matrix Computations, Part II |
| | 15.25 - 15.35: | discussion |
| 15.35 | **closure, tea, departure** | |

# numerical excellence in software

## NUMERICAL

- **NAG Fortran Library** - A collection of over 1165 user-callable subroutines for mathematical and statistical computation.

- **NAG Fortran Foundation Library** - A substantial subset of the full Fortran Library for use on PCs and Workstations where workspace is at a premium.

- **NAG C Library** - highly sophisticated problem-solvers for the growing C population in the workplace, universities and laboratories.

- **NAG Ada Library** - Reusable Ada packages, giving both generic templates and predefined standard instances, for numerical applications.

- **NAG Pascal Library (Turbo)** - Procedures covering a wide range of problems in numerical computation.

## SYMBOLIC

- **Axiom** - A powerful symbolic solver complete with a high level interactive language, a comprehensive user extensible library and a visualisation tool for the manipulation of graphical output. Supported by a powerful hypertext system.

## STATISTICAL

- **Genstat** - The powerful interactive general statistics package with a menu system, high quality graphics and extensive programming facilities.

- **Glim** - A flexible, interactive package for fitting linear models and performing data exploration.

## VISUALISATION

- **Graphics Library** - A stand-alone Fortran subroutine library which generates graphical output, and interfaces to many different plotting packages. Can also be used in conjunction with the numerical libraries.

- **IRIS Explorer** - A powerful yet easy-to-use sophisticated visualisation system with a user environment that allows users and application developers to build complex applications for visualising sets of data.

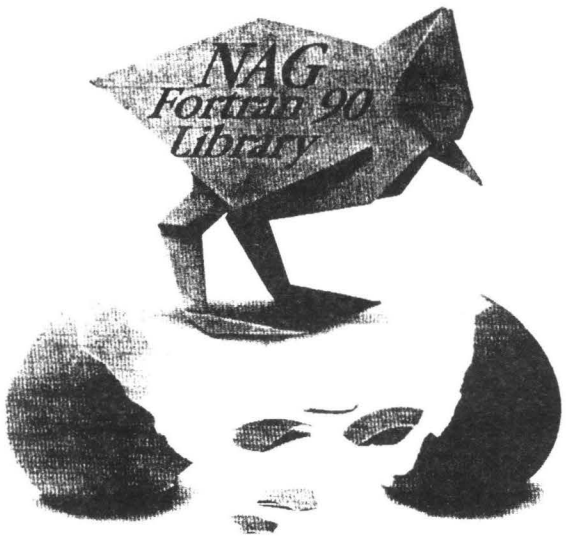
## NAGWARE

- **f90 Compiler** - A full ISO/ANSI standard Fortran 90 Compiler.

- **FTN90 Compiler** - A full ISO/ANSI standard Fortran 90 Compiler for PCs. (Note: A student level version is also available.)


Stationsplein 47
5611 BC Eindhoven
Tel 040-434957
Fax 040-434462

# Get into Fortran 90 with our new baby.




## *Introducing NAG **fl90** the world's first Fortran 90 numerical procedure library*

NAG is pleased to announce the release of the world's first Fortran 90 numerical procedure library. NAG fl90 has been designed from the outset to capitalise on the increased functionality, power and simplicity of Fortran 90 - the new Fortran standard.

The company that brought you the world's first Fortran 90 Compiler and development tools, now provide a complete suite of software for scientific computation in Fortran 90. The NAG fl90 library combines powerful numerics with 90s code - this is our investment in your future.

For further information about our range of libraries, compilers and tools for software developers and programmers contact us now at the address below:

Official distributor in the Benelux

**Computing & Systems Consultants bv**

NAG®

Stationsplein 47
5611 BC Eindhoven
Tel: 040-434957
Fax: 040-434462
email: alex@csc.mhs.compuserve.com

# Parallel Solution of the Incompressible Navier-Stokes Equations

Erik Brakkee, Guus Segal, Kees Kassels
Applied Analysis Group
Department of Mathematics
Delft University of Technology

## Abstract

In recent years, much progress has been made in the application of computational fluid dynamics to engineering problems. Efficient methods have been developed for solving flow problems in regular geometries where the subdomain boundaries coincide with the external boundaries of the flow domain.

To extend these methods to complex geometries, domain decomposition techniques are used. An example of such a complex geometry is the often occurring problem of computing the flow around an object.

The pressure-correction algorithm which is currently used for the solution of the incompressible Navier-Stokes equations has been extended to complex geometries using a domain decomposition approach combined with a Krylov subspace acceleration procedure. The domain decomposition algorithm tries to retain the discretized equations across the interface. In this way we obtain both global momentum continuity and global conservation of mass. The domain decomposition approach has proven to be capable of accurately handling complex geometries.

As flow problems become larger and more geometrically complex, the amount of work needed to solve such problems increases rapidly. Therefore, there is much need for greater computing power. An important method for achieving this goal is a parallel implementation of the domain decomposition algorithm. Besides a speedup of the computation, the distribution of the workload on several processors can decrease both the number of computations per processor and the

amount of memory required per processor. This approach enables us to solve problems on a cluster of workstations that well exceed the computing power of only a single such workstation.

We have developed a parallel version of the ISNaS incompressible code on a cluster of workstations using the PVM software system. Reasonable speedups of a factor 2-4 for two-dimensional problems have been obtained even with the relatively slow ethernet network. However, as experiments show, efficiency deteriorates rapidly as the number of participating workstations increases. This is particularly surprising since large amounts of work are done between consecutive communications steps.

This talk is devoted to explain these observations.

References:

A. Segal, P. Wesseling, J.J.I.M. van Kan, C.W. Oosterlee and C.G.M. Kassels,
Invariant discretization of the incompressible Navier-Stokes equations in boundary fitted coordinates,
Intern. J. Numer. Meth. Fluids 15 (1992), pp. 411-426.

E. Brakkee and A. Segal,
A Parallel Domain Decomposition Methods for the Incompressible Navier-Stokès Equations,
Proc. Intern. EUROSIM Conf. on Massively Parallel Processing, Applications and Development, 21-23 June 1994, Delft Univ. of Techn.

Clemens H. Cap and Volker Strumpen,
Efficient parallel computing in distributed workstation environments,
Parallel Computing 19 (1993), pp. 1221-1234.

# Multistep Methods for Differential-Algebraic Equations in Euler-Lagrange Form

Claus Führer Edda Eich

In this paper we will give an introduction to "classical" implementations of multistep methods to differential-algebraic equations. First we discuss properties and implementational aspects for semi-explicit index-1 problems. Then we will consider index-2 and index-3 problemes. Typically in mechanical systems differential–algebraic systems occur in index-3 form. We will see that, while "formally" applicable, multistep methods cause numerical problems when applied directly to this class of equations. We will present techniques to reformulate the equations of mechanical motion in such a way, that these problems can be omitted. The third part of the talk will cover some special aspects of solving DAEs in mechanical system simulation.

## 1 Multistep methods for semi-explicit index-1 equations

Semi-explicit index-1 DAEs have the following form:

$$\dot{x} = f(x,\lambda) \tag{1.1a}$$
$$0 = g(x,\lambda) \tag{1.1b}$$

with the $n_\lambda \times n_\lambda$-matrix

$$\frac{\partial g}{\partial \lambda}(x,\lambda)$$

being nonsingular in a neighbourhood of the solution. By introducing a small parameter $\epsilon$ this equation can be seen as the limit for $\epsilon \to 0$ of the singular perturbed ODE :

$$\dot{x} = f(x,\lambda) \tag{1.2a}$$
$$\epsilon\dot{\lambda} = g(x,\lambda) \tag{1.2b}$$

Discretizing this equation by a multistep scheme and letting $\epsilon = 0$ results in

$$\sum_{i=0}^{k} \alpha_i x_{n+1-i} = h \sum_{i=0}^{k} \beta_i f(x_{n+1-i}, \lambda_{n+1-i}) \tag{1.3a}$$

$$0 = \sum_{i=0}^{k} \beta_i g(x_{n+1-i}, \lambda_{n+1-i}) \tag{1.3b}$$

Herein is $h$ the step-size and $\alpha_i, \beta_i$ are method dependent coefficients.

Equation (1.2) can be viewed as an implicit algebraic equation for $x_{n+1}$ and $\lambda_{n+1}$, which must be solved at every time step $t_{n+1}$. For BDF methods ($\beta_i = 0, i = 1, \ldots, k$) this coincides with an alternative formulation, which has no longer a relation to singular perturbed problems:

$$\sum_{i=0}^{k} \alpha_i x_{n+1-i} = h \sum_{i=0}^{k} \beta_i f(x_{n+1-i}, \lambda_{n+1-i}) \tag{1.4a}$$

$$0 = g(x_{n+1-i}, \lambda_{n+1-i}) \tag{1.4b}$$

After discussing the convergence properties of multistep methods for this class of equations we will cover the following more pratical questions:

- Error- and stepsize control
- Solution of the implicit equation, i.e. implementation of Newton's method
- Order Control
- Special aspects, i.e. smooth integration and treatment of discontinuities.

# 2 Euler-Lagrange Equations

One large field of applications, where DAEs occur, are the equations of motion of constrained rigid mechanical systems. After a brief overview over current multibody formalisms and the form of DAEs they generate, we will discuss the numerical treatment of these equations. They have the following structure:

$$\dot{p} = v \tag{2.1a}$$

$$M(p)\dot{v} = f(p, v) - G(p)^T \lambda \tag{2.1b}$$

$$0 = g(p) \tag{2.1c}$$

with $p$ being the position variables, $v$ the velocity variables, $G(p) := \frac{\partial g}{\partial p}(p)$ the constraint matrix and $\lambda$ the Lagrange multipliers, which force the motion $p, v$ to meet the constraint. This is an index-3 DAE. We will discuss convergence

of multistep methods for this class of problems and point out implementation problems. These problems make it necessary to reduce the index of the problem in some way. Classically, this is done by formulating the constraint on velocity level (index-2)

$$0 = G(p)v$$

or on acceleration level (index-1)

$$0 = G(p)\dot{v} + \frac{\partial}{\partial p}(G(p)v)\dot{p}$$

Reformulating the mechanical problem by reducing its index causes so-called drift-off effects, i.e. the stability of the problem is affected by the underlying differentiation. These instabilities can be removed by stabilized formulations and projection methods. We will present various alternatives in stabilizing the index-reduced equations and discuss their pro and contras when applying these techniques to Euler Lagrange equations. We will briefly present the follwoing stabilized formulations

- state-space related approaches
- the introduction of additional Lagrange multipliers
- coordinate projection methods

# 3 Special Aspects

In mechanical problems, mainly in vehicle dynamics, DAEs often occur in the context of contact problems. A typical application is the simulation of wheel rail systems. We will show an example for impasse points and discuss modelling alternatives in these cases. Another special problem occurs, when solving DAEs is part of an optimization problem, like in parameter identification of constrained mechanical system.

# References


[1] C. Arévalo. Matching the structure of DAEs and multistep methods. Ph.D. dissertation, Lund University 1993.

[2] A. Barrlund. Constrained least squares methods, of implicit type, for the numerical solution to higher index linear variable coefficient differential algebraic systems. Technical Report UMINF-166.89, University of Ume, Institute of Information Processing, Sweden, 1989.

[3] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Meth. in Appl. Mechanics*, 1:1 – 16, 1972.

[4] H. Bock, E. Eich, and J. Schlöder. Numerical solution of constrained least squares boundary value problems in differential-algebraic equation. In Strehmel, editor, *Proceedings NUMDIFF 87*. Teubner, Halle, 1987.

[5] H. Brandl, R. Johanni, and M. Otter. A Very Efficient Algorithm for the Simulation of Robots and Similar Multibody Systems Without Inversion of the Mass Matrix. In *Proc. IFAC/IFIP/IMACS International Symposium on The Theory of Robots, Vienna, Austria*, Dec 1986.

[6] K. Brenan, S. Campbell, and L. Petzold. *The Numerical Solution of Initial Value Problems in Ordinary Differential-Algebraic Equations*. North Holland Publishing Co., 1989.

[7] L. Chua and A. Deng. Impasse points. part i: Numerical aspects. *Intern. J. of Circuit Theory and Applications*, 17:213–35, 1989.

[8] E. Eich. *Projizierende Mehrschrittverfahren zur numerischen Lösung der Bewegungsgleichungen technischer Mehrkörpersysteme mit Zwangsbedingungen und Unstetigkeiten*. PhD thesis, Institut für Mathematik, Universität Augsburg, 1991. also as "VDI-Fortschrittsberichte", VDI-Verlag, Düsseldorf, 1992.

[9] E. Eich. Convergence results for a coordinate projection methods applied to constrained mechanical systems. to appear in SIAM J. Numer. Anal., 1992.

[10] E. Eich, C. Führer, B. Leimkuhler, and S. Reich. Stabilization and Projection Methods for Multibody Dynamics. Technical report, Helsinki University of Technology, Finland, 1990.

[11] E. Eich, C. Führer, and J. Yen. On the error control for multistep methods applied to odes with invariants and daes in multibody dynamics. Technical Report R-160, University of Iowa, 1993.

[12] C. Führer. *Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen*. PhD thesis, Mathematisches Institut, Technische Universität München, 1988.

[13] C. Führer and B. Leimkuhler. A new class of generalized inverses for the solution of discretized Euler – Lagrange equations. In E. Haug and R. Deyo, editors, *NATO Advanced Research Workshop on Real – Time Integration Methods for Mechanical System Simulation*. Springer, Heidelberg, 1990.

[14] C. Führer and B. Leimkuhler. Numerical solution of differential-algebraic equations for constrained mechanical motion. *Numerische Mathematik*, 59:55–69, 1991.

[15] C. Führer and R. Schwertassek. Generation and Solution of Multibody System Equations. *Int. Journal of Nonl. Mechanics*, 25(2/3):127–141, 1990.

[16] C. Führer and O. Wallrapp. A computer-oriented method for reducing linearized multibody equations by incorporating constraints. *Comp. Meth. Apl. Mech. Eng.*, 46:169 – 175, 1984.

[17] C. Gear. Simultaneous numerical solution of differential-algebraic equations. *IEEE Trans. Circuit Theory*, CT-18(1):89 – 95, 1971.

[18] C. Gear. Maintaining solution invariants in the numerical solution of ODE. *SIAM J. Sci. Stat. Comput.*, 7(3):734–743, 1986.

[19] C. Gear, G. Gupta, and B. Leimkuhler. Automatic Integration of the Euler-Lagrange Equations with Constraints. *J. Comp. Appl. Math.*, 12 & 13:77–90, 1985.

[20] C. W. Gear and L. R. Petzold. Ode methods for the solution of differential algebraic systems. *SIAM J. Numer. Anal.*, 21:716–728, 1984.

[21] E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment.* Teubner-Texte zur Mathematik No. 88. BSB B.G. Teubner Verlagsgesellschaft, Leipzig, 1986.

[22] W. Grünhagen. *Zur Stabilisierung der numerischen Integration von Bewegungsgleichungen.* PhD thesis, Universität Braunschweig, 1979.

[23] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Equations by Runge-Kutta Methods.* Lecture Notes in Mathematics Vol. 1409. Springer, Heidelberg, 1989.

[24] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential- Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[25] E. Haug and J. Yen. Implicit numerical integration of constrained equations of motion via generalized coordinate partitioning. In E.Haug and R.Deyo, editors, *NATO Advanced Research Workshop on Real – Time Integration Methods for Mechanical System Simulation.* Springer, Heidelberg, 1990.

[26] B. Leimkuhler. Error estimates for differential-algebraic equations. Technical Report UIUCDCS-R-86-1287, Dept. of Comp. Science, Univ. of Illinois, 1988.

[27] B. Leimkuhler, L. Petzold, and C.W.Gear. Approximation methods for the consistent initialization fo differential-algebraic equations. *SIAM Journal on Numerical Analysis*, submitted for publication.

[28] P. Lötstedt. Analysis of some difficulties encountered in the simulation of mechanical systems with constraints. Technical report, The Royal Institute of Technology, Stockholm, 1979.

[29] P. Lötstedt. A numerical method for the simulation of mechanical systems with unilateral constraints. Technical report, The Royal Institute of Technology, Stockholm, 1979.

[30] P. Lötstedt. On the relation between singular perturbation problems and differntial-algebraic equations. Technical report, Uppsala University, 1985.

[31] P. Lötstedt and L. Petzold. Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints I: Convergence Results for BDF. *Math.Comp.*, 46:491–516, 1986.

[32] C. Lubich, U. Nowak, U.Pöhle, and C. Engstler. Differential-algebraic equations for constrained mechanical systems and their numerical solution by extrapolation methods. In *Problemes Nonlineaires Appliques: Systemes Algebro-Differentiels*, pages 20–42. INRIA,EDF, Clamard, France, 1992.

[33] R. März. Multistep methods for initial value problems in implicit differential-algebraic equations. *Beitr. Numer. Math.*, 12:107–123, 1984.

[34] R. März. Once more on the backward differentiation formulas applied to higher index differential-algebraic equations. *Z. Angew. Math. Mech.*, 69:T37–T39, 1989.

[35] S.E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. SIAM J Sci. Comput. 14 (1993), 677–692.

[36] T. Mrziglod. Zur Theorie und numerischen Realisierung von Lösungsmethoden bei Differentialgleichungen mit angekoppelten algebraischen Gleichungen. Master's thesis, Universität Köln, Math. Institut, 1987.

[37] P. Nikravesh. Some methods for dynamical analysis of constrained mechanical systems: A survey. In J. Haug, editor, *Computer Aided Analysis and Optimization of Mechanical System Dynamics*, pages 223–259. Springer, 1984.

[38] L. Petzold. A description of DASSL: A differential/ algebraic system solver. In *Proc. 10th IMACS World Congress, August 8-13 Montreal 1982*, 1982.

[39] L. Petzold and P. Lötstedt. Numerical solution of nonlinear differential equations with algebraic constraints ii. practical implications. *SIAM J. Sci. Stat. Comp.*, 7:720–733, 1986.

[40] F. Potra and W. Rheinboldt. On the numerical integration for Euler-Lagrange equations via tangent space parametrization. *J. Mechanics of Structure and Machines*, 19(1), 1991.

[41] F. Potra and J. Yeng. Implicit numerical integration for euler-lagrange equations via tangent space parametrization. Technical Report R-56, College of Engineering, University of Iowa, Iowa City, Iowa 52242, 1989.

[42] W. Rheinboldt. Differential - algebraic systems as differential equations on manifolds. *Math. Comp.*, 43(168):2473–482, 1984.

[43] W. Schiehlen, editor. *Multibody Handbook*. Springer, Heidelberg, to appear 1989.

[44] E. Seitz. Integration impliziter Differentialgleichungen durch Adams-Verfahren mit festem Führenden Koeffizienten. Technical Report IB 515-89-09, DLR Oberpfaffenhofen, D-8031 Wessling, 1989.

[45] L. Shampine. Conservation laws and the numerical solution of odes. Technical Report 84-1241, Sandia National Laboratories, Livermore, 1984.

[46] B. Simeon, C. Führer, and P. Rentrop. Differential-algebraic equations in vehicle system dynamics. *Surv. Math. Ind.*, 1:1–37, 1991.

[47] B. Simeon, C. Führer, and P. Rentrop. The Drazin inverse in multibody system dynamics. *Numerische Mathematik*, 1993.

[48] B. Simeon. Numerische Integration mechanischer Mehrkörpersysteme: Projizierende Deskriptorformen, Algorithmen und Rechenverfahren, Dissertation TU M—nchen, 1994.

[49] G. Söderlind. DASP3-A program for the numerical integration of partitioned stiff ODE:s and differential-algebraic systems. Technical Report TRITA-NA-8008, Institute for Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, 1980.

*Address:*

DR. CLAUS FÜHRER, Institute for Dynamics and Robotics, German Aerospace Research Est. (DLR), Postfach 116, D-82230 Wessling, Germany, e-mail: claus.fuehrer@dlr.de

DR. EDDA EICH, Linde AG, D-82409 H˜llriegelskreuth, e-mail: na.eich@na-net.ornl.gov

# $\beta$-blocked Multistep Methods for Euler-Lagrange DAEs

Claus Führer  Carmen Arévalo  Gustaf Söderlind

Many different methods have been suggested for the numerical solution of index 2 and 3 Euler-Lagrange equations. We focus on 0-stability of multistep methods and investigate the relations between some well-known computational techniques. By various modifications, referred to as $\beta$-blocking, some basic shortcomings of multistep methods may be overcome. This approach is related to projection techniques and has a clear and well-known analogy in control theory. In particular, it is not necessary to use BDF methods for the solution of high index problems; indeed "nonstiff" methods may be used for part of the system provided that the state-space form is nonstiff. We give a simplified multibody model of a truck to illustrate the techniques and to demonstrate the results.

We restrict ourselves to a special class of DAEs of the form

$$\begin{aligned} \dot{x} &= f(x) - G^T(x)\lambda \\ 0 &= g(x) \end{aligned}$$

where the algebraic variable $\lambda$ occurs linearly, $G(x) = \partial g/\partial x$ and $G(x)G^T(x)$ is invertible. An important class of practical problems leading to DAEs of this form are constrained multibody systems in index-2 form:

$$\begin{aligned} \dot{p} &= v \\ M\dot{v} &= f(p,v) - G^T(p)\lambda \\ 0 &= G(p)v \end{aligned}$$

with $p, v$ being the position and velocity variables, $f(p,v)$ the applied forces, $M$ a positive definite mass matrix, $G(p)$ the constraint matrix, and $\lambda$ the Lagrange multipliers.

## Linear Analysis

In the first part of the talk we will discuss multistep methods for Euler-Lagrange Equations and motivate the following alternative discretization scheme of the

problem:

$$\begin{aligned} \rho x_n &= h\sigma A x_n - hG^T \lambda_n \\ 0 &= G x_n \end{aligned}$$

with $\rho$ and $\sigma$ being the multistep difference operators defined by

$$\rho x_n := \sum_{i=0}^{k} \alpha_i x_{n-i} \qquad \text{and} \qquad \sigma x_n := \sum_{i=0}^{k} \beta_i x_{n-i}$$

This discretization differs from the "classical" approach as different discretizations are applied to different parts of an equation. This formulation avoids instabilities caused by structural eigenvalues of the propagation matrix of the method and allows also the application of Adams-Moulton methods to index-2 problems. Shifting these eigenvalues into the origin results in this scheme, which is called $\beta$- blocked multistep method.

## General case

This method can be generalized to the nonlinear case in various ways. We study here the approach

$$\rho(x_n) = h\sigma(f(x_n) - G(x_n)^T \lambda_n) + hG(x_n)\tau\lambda_n \tag{1}$$

$$0 = g(x_n) \tag{2}$$

with

$$\tau x_n = \sum_{i=0}^{k} \gamma_i x_{n-i},$$

where the polynomial $\sigma - \tau$ satisfies the strict root condition. This requirement is not sufficient for consistency and a high order. We have to ask in addition for

$$\tau x = O(h^k)$$

where $k$ is the order the method given by $\rho$ and $\sigma$ when applied to explicit ODEs.

Unfortunately when taking $\rho$ and $\sigma$ to be the Adams-Moulton operators this condition together with the strict root condition cannot be met for orders greater than 4.

We will present a method which shares a lot of the properties of Adams-Moulton methods but ideally fulfills the above give conditions. This method is the difference corrected BDF method first introduced in [11].

## Example

The ideas are presented on a simple multibody model of a truck.

## References


[1] J. Åström and Bjorn Wittenmark. *Computer-Controlled Systems, Theory and Design.* Prentice-Hall, Englewood Cliffs, N.J, 1990.

[2] C. Arévalo. Matching the structure of DAEs and multistep methods. Ph.D. dissertation, Lund University 1993.

[3] C. Arévalo, Claus Führer and Gustaf Söderlind. $\beta$-blocked Multistep Methods for Euler-Lagrange DAEs: I. Linear Case Report, Lunds Universitet, Institutet för Datalogi och Numerisk Analys, 1994.

[4] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *The Numerical Solution of Initial Value Problems in Ordinary Differential-Algebraic Equations.* North Holland Publishing Co., 1989.

[5] V. Brasey and E. Hairer. Half - explicit Runge-Kutta methods for differential - algebraic systems of index 2. SIAM J.Numer. Anal. 30(1993), p. 538ff.

[6] E. Eich, C. Führer, B. Leimkuhler, and S. Reich. Stabilization and Projection Methods for Multibody Dynamics. Technical report, Helsinki University of Technology, Finland, 1990.

[7] E. Eich and C. Führer. Numerical methods in multibody dynamics. Technical Report IB 515-92-17, DLR Institut Dynamik der Flugsysteme, D-82230 Wessling, 1992.

[8] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential- Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[9] S.E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. SIAM J Sci. Comput. 14 (1993), 677–692.

[10] Ch. Lubich. $h^2$ extrapolation methods for differential-algebraic equations of index-2. *Impact Comp. Sci. Eng.*, 1:260–268, 1989.

[11] G. Söderlind. A multipurpose system for the numerical Integration of ODEs *Appl.Math.Comp.*, (31) p. 346 ff, 1986.



*Address:*
DR. CARMEN ARÉVALO, Dept. de Matemáticas Puras y Aplicadas, Universidad Simón Bolívar, Apartado 89000, Caracas 1080-A, Venezuela, e-mail: camena@cesma.usb.ve

DR. CLAUS FÜHRER, Institute for Dynamics and Robotics, German Aerospace Research Est. (DLR), Postfach 116, D-82230 Wessling, Germany, e-mail: claus.fuehrer@dlr.de
PROF.DR. GUSTAF SÖDERLIND, Dept. of Computer Sciences and Numerical Analysis, Lund University, Box 118, S-22100 Lund, Sweden, e-mail: gustaf@dna.lth.se

# Runge-Kutta-Type Methods for Differential-Algebraic Systems

Ernst Hairer, Geneva, Switzerland

## 1. Examples of Differential-Algebraic Problems

We consider initial value problems of the from

$$Bu' = F(u), \qquad u(0) = u_0, \tag{1.1}$$

where $B$ is a constant square matrix. If $B$ is invertible, then (1.1) represents an ordinary differential equation (ODE). Otherwise, we are concerned with a differential-algebraic equation (DAE).

**Semi-Explicit Index 1 problems.** Suppose that the matix $B$ in Eq. (1.1) is diagonal with entries 1 and 0. Then, with $u = (y, z)$, the problem can be written in the form

$$\begin{aligned} y' &= f(y, z), \qquad y(0) = y_0 \\ 0 &= g(y, z), \qquad z(0) = z_0. \end{aligned} \tag{1.2}$$

If the initial values satisfy $g(y_0, z_0) = 0$ and if

$$g_z(y_0, z_0) \qquad \text{is invertible,} \tag{1.3}$$

then, by the implicit function theorem, we can solve the equation $g(y, z) = 0$ for $z$ (in a neighbourhood of the initial value) and obtain $z = G(y)$. Inserted into the first equation of (1.2) we obtain an ordinary differential equation. Standard ODE-theory can be applied to prove existence and uniqueness of a solution.

**Semi-Explicit Index 2 Problems.** In the system

$$\begin{aligned} y' &= f(y, z), \qquad y(0) = y_0 \\ 0 &= g(y), \qquad z(0) = z_0 \end{aligned} \tag{1.4}$$

the variable $z$ can be interpreted as control variable and has to be determined in such a way that the solution $y(t)$ of the differential equation satisfies the constraint $g(y) = 0$. Obviously, the condition (1.3) is not satisfied. Differentiating once the constraint, we obtain $0 = g_y(y)y' = g_y(y)f(y, z)$. In the case that

$$g_y(y_0)f_z(y_0, z_0) \qquad \text{is invertible,} \tag{1.5}$$

the relation $0 = g_y(y)f(y, z)$ can be solved for $z$ and we again obtain an ODE. Consequently, if the initial values are consistent, i.e., if

$$g(y_0) = 0, \qquad g_y(y_0)f(y_0, z_0) = 0, \tag{1.6}$$

the system (1.4) has a locally unique solution.

**Constrained Mechanical Systems.** An interesting class of differential-algebraic systems appears in the mechanical modeling of constraint systems ($q \in \mathbb{R}^n$, $v \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$, $m < n$):

$$q' = v \tag{1.7a}$$

$$M(q)v' = f(q, v) - G^T(q)\lambda \tag{1.7b}$$

$$0 = g(q) \tag{1.7c}$$

where $G(q) = g_q(q)$. Here we have to differentiate the constraint twice in order to be able to write the algebraic variable $\lambda$ in terms of $q$ and $v$:

$$0 = G(q)v \tag{1.7d}$$

$$0 = g_{qq}(q)(v,v) + G(q)v'. \tag{1.7e}$$

Indeed, Eqs. (1.7b) and (1.7e) constitute a linear system

$$\begin{pmatrix} M(q) & G^T(q) \\ G(q) & 0 \end{pmatrix} \begin{pmatrix} v' \\ \lambda \end{pmatrix} = \begin{pmatrix} f(q,v) \\ -g_{qq}(q)(v,v) \end{pmatrix} \tag{1.8}$$

for $v'$ and $\lambda$. If the matrix appearing in Eq. (1.8) is invertible, if the initial values satisfy

$$0 = g(q_0), \qquad 0 = G(q_0)v_0, \tag{1.9}$$

and if $\lambda_0$ is such that (1.8) is verified, then we have a locally unique solution of the problem (1.7).

## 2. Survey on Numerical Approaches

During the last years there has been much progress in the development and analysis of numerical methods for DAEs.

The numerical solution of index 1 problems (1.2) is relatively easy. Since, by assumption (1.3), the algebraic constraint is equivalent to $z = G(y)$, we can solve $y' = f\big(y, G(y)\big)$ by any ODE-method. Each function evaluation requires the numerical solution of the nonlinear system $0 = g(y,z)$.

**Index Reduction + Projection.** For higher index systems the idea is the following: differentiate the algebraic constraints until an index 1 system is obtained (so that the algebraic variables can be expressed in terms of the differential variables) and integrate numerically the resulting index 1 system. E.g., in order to solve the system (1.7a,b,c) we consider the ODE

$$q' = v, \qquad v' = v'(q,v), \tag{1.10}$$

where $v'(q,v)$ is the solution of the linear system (1.8). If the initial values are consistent (i.e., Eq. (1.9) is verified), the exact solutions of (1.10) and of (7a,b,c) are the same.

However, due to the discretization error of the numerical method, the constraints (1.7c) and (1.7d) will no longer be satisfied by the numerical solution ("drift-off"). It is therefore highly recommended to project the numerical solution after every integration step back to the manifold defined by $g(q) = 0$ and $G(q)v = 0$.

**State-Space Form.** The idea is to work with minimal coordinates describing the solution manifold. Consider, for example, the constrained mechanical system (1.7). We fix $(n-m) \times n$-dimensional matrices $A_1$ and $A_2$ such that the following two systems possess a locally unique solution $q = \phi(\widetilde{q})$, $v = \psi(\widetilde{q}, \widetilde{v})$ (see [Po], [PoR]):

$$\begin{pmatrix} g(q) \\ A_1^T(q - q_0) \end{pmatrix} = \begin{pmatrix} 0 \\ \widetilde{q} \end{pmatrix}, \qquad \begin{pmatrix} G(q)v \\ A_2^T(v - v_0) \end{pmatrix} = \begin{pmatrix} 0 \\ \widetilde{v} \end{pmatrix}. \tag{1.11}$$

This yields a parametrization of the manifold defined by $g(q) = 0$, $G(q)v = 0$. In the new coordinates $\widetilde{q}$, $\widetilde{v}$ the problem (1.7) becomes

$$\widetilde{q}\,' = A_1^T \psi(\widetilde{q}, \widetilde{v}), \qquad \widetilde{v}\,' = A_2^T v'\big(\phi(\widetilde{q}), \psi(\widetilde{q}, \widetilde{v})\big), \tag{1.12}$$

where $v'(q,v)$ is the solution of the linear system (1.8). This is an ODE which can be solved numerically and gives approximations to $\widetilde{q}$, $\widetilde{v}$ and hence also to $q$, $v$.

**Direct Application of ODE-Methods.** If the matrix $B$ in Eq. (1.1) is invertible, then (1.1) represents an ordinary differential equation $u' = B^{-1}F(u)$ and standard ODE-methods can be applied. For example, a Runge-Kutta method becomes

$$U_i = u_0 + h\sum_{j=1}^{s} a_{ij}k_j, \qquad i = 1, \ldots, s \tag{1.13a}$$

$$u_1 = u_0 + h\sum_{i=1}^{s} b_i k_i \tag{1.13b}$$

where

$$k_i = B^{-1}F(U_i), \qquad i = 1, \ldots, s. \tag{1.14}$$

If the matrix $B$ is singular, we simply replace the condition (1.14) by

$$Bk_i = F(U_i), \qquad i = 1, \ldots, s. \tag{1.15}$$

Eq. (1.15) with $U_i$ inserted from (1.13a) represents a nonlinear system for $k_1, \ldots, k_s$. For problems of the form (1.2), (1.4) or (1.7) this nonlinear system can be shown to have a unique solution, if the Runge-Kutta matrix $(a_{ij})$ is invertible. The convergence of these methods (order reduction!) is nowadays well understood (see e.g., [HaLR] or [HaW], Chap. VI). *Stiff accuracy* of the method, i.e.,

$$a_{si} = b_i \qquad \text{for} \qquad i = 1, \ldots, s \tag{1.16}$$

is an important hypothesis for these convergence properties.

**Special Methods for Special Problems.** In order to increase their efficiency, methods can be adapted to the structure of special classes of DAEs. An interesting example will be presented in the next section.

## 3. Half-Explicit Runge-Kutta Methods for Index 2 Problems

A disadvantage of the direct application of ODE-methods (see Eq. (1.13)) is that it requires implicit methods. If the problem is in semi-explicit form, such as the index 2 system (1.4), then it is natural to look for methods which are explicit in the differential variable $y$ and implicit in the algebraic variable $z$ only.

**Definition of Half-Explicit Runge-Kutta Methods.** In the context of RK-methods and index 2 problems (1.4) we are led to the following method ([HaLR], [Br]):

$$Y_i = y_0 + h\sum_{j=1}^{i-1} a_{ij}f(Y_j, Z_j), \qquad i = 1, \ldots, s \tag{1.17a}$$

$$0 = g(Y_i), \qquad i = 1, \ldots, s \tag{1.17b}$$

$$y_1 = y_0 + h\sum_{i=1}^{s} b_i f(Y_i, Z_i) \tag{1.17c}$$

$$0 = g(y_1). \tag{1.17d}$$

The initial value is assumed to satisfy $g(y_0) = 0$. This method is then applied as follows: for $i = 1$, Eq. (1.17a) defines $Y_1 = y_0$ and (1.17b) is automatically satisfied for $i = 1$. If we insert $Y_2$ from (1.17a) into (1.17b), we obtain $0 = g\big(y_0 + ha_{21}f(Y_1, Z_1)\big)$, which is a nonlinear equation for $Z_1$. Once $Z_1$ is computed, (1.17a) constitutes an explicit formula for $Y_2$ and the procedure can be repeated for the further stages.

**Properties of the Method.** In order that the arising nonlinear equations possess a solution, we have to require that

$$a_{i,i-1} \neq 0 \quad \text{for } i = 2, \ldots, s \quad \text{and} \quad b_s \neq 0. \tag{1.18}$$

The error propagation is very similar to that for ODE-methods. Studying the local error, one observes that the usual order conditions for RK-methods are necessary but not sufficient in order to get $y_1 - y(t_0 + h) = \mathcal{O}(h^{p+1})$. Additional order conditions have to be imposed.

**Application to Constrained Mechanical Systems.** Instead of considering the system (1.7a,b,c) we apply our method (1.17) to the index 2 system (1.7a,b,d). This yields

$$Q_i = q_0 + h \sum_{j=1}^{i-1} a_{ij} V_j, \qquad V_i = v_0 + h \sum_{j=1}^{i-1} a_{ij} V'_j, \tag{1.19}$$

where

$$\begin{pmatrix} M(Q_i) & G^T(Q_i) \\ G(Q_{i+1}) & 0 \end{pmatrix} \begin{pmatrix} V'_i \\ \Lambda_i \end{pmatrix} = \begin{pmatrix} f(Q_i, V_i) \\ -G(Q_{i+1})\big(v_0 + h \sum_{j=1}^{i-1} a_{i+1,j} V'_j\big)/(h a_{i+1,i}) \end{pmatrix}$$

and an analoguous formula for $q_1, v_1$. Since the constraint (1.7d) depends linearly on $v$, nonlinear systems are completely avoided. Promising numerical results of a 5th order method are reported in [Br2].

## References


[Br] V. Brasey (1992): *A half-explicit Runge-Kutta method of order 5 for solving constrained mechanical systems*. Computing **48**, 191-201.

[Br2] V. Brasey (1994): *Half-eplicit methods for semi-explicit differential-algebraic equations of index 2*. Thèse, Section de Mathématiques, Université de Genève, 144 pp.

[BrH] V. Brasey & E. Hairer (1993): *Half-explicit Runge-Kutta methods for differential-algebraic systems of index 2*. SIAM J. Numer. Anal. **30**, 538-552.

[BrCP] K. Brenan, S. Campbell & L. Petzold (1989): *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland.

[FüL] C. Führer & B. Leimkuhler (1991): *Numerical solution of differential-algebraic equations for constrained mechanical motion*. Numer. Math. **59**, 55-69.

[GrM] E. Griepentrog & R. März (1986): *Differential-Algebraic Equations and Their Numerical Treatment*. Teubner-Texte zur Mathematik, Band **88**.

[HaLR] E. Hairer, Ch. Lubich & M. Roche (1989): *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Springer Lecture Notes in Mathematics 1409.

[HaW] E. Hairer & G. Wanner (1991): *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics **14**, Springer-Verlag.

[Po] F. Potra (1993): *Runge-Kutta integrators for multibody dynamics*. To be published.

[PoR] F. Potra & W. Rheinboldt (1991): *On the numerical solution of the Euler-Lagrange equations*. J. Mechanics of Structures and Machines **19**, 1-18.

# Long-Time Integration of Constrained Hamiltonian Systems

Ernst Hairer, Geneva, Switzerland

## 1. Formulation of the Problem

We consider constrained mechanical systems where dissipative forces are neglected. Such systems can be written in the form

$$q' = H_p(q,p) \tag{2.1a}$$

$$p' = -H_q(q,p) - G^T(q)\lambda \tag{2.1b}$$

$$0 = g(q). \tag{2.1c}$$

Here $q = (q_1, \ldots, q_n)$ are generalized coordinates, $p = (p_1, \ldots, p_n)$ are generalized momenta, $\lambda = (\lambda_1, \ldots, \lambda_m)$ are Lagrange multipliers and $H(q,p)$ represents the Hamiltonian of the system. Differentiation of the constraint shows that the solutions of (2.1) lie on the manifold

$$\mathcal{M} = \{(q,p) \mid 0 = g(q),\ 0 = G(q)H_p(q,p)\}. \tag{2.2}$$

The formulation (2.1) has the advantage that the flow possesses interesting geometrical properties.

**Symplectic Structure.** It can be shown that the flow of (2.1) (of course, restricted to the manifold $\mathcal{M}$) is a symplectic transformation, i.e., it preserves the differential 2-form

$$\omega^2 = \sum_{k=1}^{n} dq_k \wedge dp_k. \tag{2.3}$$

This means that the sum of the projected (oriented) areas of any 2-dimensional submanifold of $\mathcal{M}$ is preserved. Another interesting property, which can easily be verified, is that the Hamiltonian function remains constant along a solution trajectory, i.e.,

$$H\big(q(t),p(t)\big) = H(q_0,p_0) \qquad \text{for all } t. \tag{2.4}$$

**Reversibility.** Typically we have $H(q,p) = p^T M^{-1}(q)p + U(q)$, so that $H_p(q,p)$ depends linearly on $p$ and $H_q(q,-p) = H_q(q,p)$. Consequently, the reflection $q \mapsto q$, $p \mapsto -p$ inverts the time direction of the flow but leaves it invariant otherwise.

More generally, assume that there exists a linear map $\rho : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ of the form $\rho(q,p) = (\rho_1(q), \rho_2(p))$ such that

$$\begin{aligned} \rho_1 H_p(q,p) &= -H_p(\rho_1 q, \rho_2 p) \\ \rho_2 H_q(q,p) &= -H_q(\rho_1 q, \rho_2 p), \end{aligned} \tag{2.5}$$

$\rho(\mathcal{M}) \subset \mathcal{M}$ and $\rho_2 G^T(q) = -G^T(\rho_1 q)\rho_3$ for some linear map $\rho_3$. Then the flow $\phi_t(q_0,p_0) = \big(q(t),p(t)\big)$ satisfies

$$\rho \circ \phi_t = \phi_t^{-1} \circ \rho. \tag{2.6}$$

## 2. Numerical Long-Time Integration

In general, numerical methods do not preserve the above-mentioned properties. This is no serious drawback for an integration over a short time interval. For long-time integration, however, it

becomes evident from numerical experiments that numerical methods preserving symplecticness and/or reversibility are superiour.

**Symplectic Runge-Kutta Methods.** For ordinary differential equations ((2.1a,b) without the constraint) it has been shown independently by Lasagni, Sanz-Serna and Suris that the numerical flow of a Runge-Kutta method is symplectic if the coefficients satisfy

$$b_i a_{ij} + b_j a_{ji} - b_i b_j = 0 \qquad \text{for all } i, j. \tag{2.7}$$

The numerical solution, obtained by such a method, can be formally interpreted as the exact solution of a perturbed Hamiltonian system (see [Ha]). Hence, all the nice properties of Hamiltonian systems are preserved.

Unfortunately, methods satisfying (2.7) cannot be stiffly accurate (see Eq. (1.16)).

**Symmetric Runge-Kutta Methods.** Denote by $\Phi_h$ the numerical flow induced by a Runge-Kutta method. Since these methods are linear, we have $\rho \circ \Phi_h = \Phi_{-h} \circ \rho$ for linear mappings satisfying the relation (2.5). Consequently, an RK-method is time reversible, i.e., it satisfies $\rho \circ \Phi_h = \Phi_h^{-1} \circ \rho$, if and only if it is symmetric (i.e., $\Phi_h^{-1} = \Phi_{-h}$). Symmetric RK-methods are characterized by

$$a_{s+1-i,s+1-j} + a_{ij} = b_j \qquad \text{for } i, j = 1, \ldots, s. \tag{2.8}$$

Suppose that our problem has a periodic solution such that one part of the orbit is mapped by $\rho$ to the other part of the orbit (e.g., pendulum). If the method is reversible, then the discretization errors in the first part will be compensated by those in the second part.

## 3. Partitioned Runge-Kutta Methods

We are interested in numerical methods for (2.1) which are symplectic, symmetric, stiffly accurate and which produce a numerical flow satisfying $\Phi_h(\mathcal{M}) \subset \mathcal{M}$. Within the class of Runge-Kutta methods this is not possible.

The idea is to apply different RK-methods to the different parts of the problem (a method with coefficients $a_{ij}, b_i$ to the $q$ variables and one with coefficients $\widehat{a}_{ij}, \widehat{b}_i$ to the $p$ variables). This yields

$$Q_i = q_0 + h \sum_{j=1}^{s} a_{ij} H_p(Q_j, P_j) \tag{2.9a}$$

$$P_i = p_0 - h \sum_{j=1}^{s} \widehat{a}_{ij} \Big( H_q(Q_j, P_j) + G^T(Q_j) \Lambda_j \Big) \tag{2.9b}$$

$$0 = g(Q_i) \tag{2.9c}$$

$(i = 1, \ldots, s)$ and

$$q_1 = q_0 + h \sum_{i=1}^{s} b_i H_p(Q_i, P_i) \tag{2.9d}$$

$$p_1 = p_0 - h \sum_{i=1}^{s} \widehat{b}_i \Big( H_q(Q_i, P_i) + G^T(Q_i) \Lambda_i \Big). \tag{2.9e}$$

**Stiff Accuracy.** Stiff accuracy is important for solving DAEs, because it implies that the numerical solution is equal to the last internal stage. Hence the algebraic constraint is automatically

satisfied. In our situation the constraint only depends on $q$. Therefore, only the method with coefficients $a_{ij}, b_i$ has to be stiffly accurate, i.e.,

$$a_{si} = b_i \qquad \text{for} \quad i = 1, \ldots, s. \tag{2.10}$$

**Symplecticness.** The method (2.9) is symplectic if ([SaSC], [Ja])

$$\begin{aligned} &\widehat{b}_i = b_i \qquad \text{for} \quad i = 1, \ldots, s, \\ &b_i \widehat{a}_{ij} + \widehat{b}_j a_{ji} - b_i \widehat{b}_j = 0 \qquad \text{for} \quad i, j = 1, \ldots, s. \end{aligned} \tag{2.11}$$

It is important that the method is applied to the index 3 formulation of the problem.

**Symmetry.** Method (2.9) is symmetric, if both RK-methods are symmetric, i.e.,

$$\begin{aligned} &a_{s+1-i,s+1-j} + a_{ij} = b_j \qquad \text{for} \quad i, j = 1, \ldots, s, \\ &\widehat{a}_{s+1-i,s+1-j} + \widehat{a}_{ij} = \widehat{b}_j \qquad \text{for} \quad i, j = 1, \ldots, s, \end{aligned} \tag{2.12}$$

**Numerical Flow on the Solution Manifold.** As a consequence of (2.10), (2.11) and (2.12) we necessarily have

$$a_{1j} = 0 \qquad \text{for} \quad j = 1, \ldots, s, \tag{2.13a}$$

$$\widehat{a}_{is} = 0 \qquad \text{for} \quad i = 1, \ldots, s, \tag{2.13b}$$

Condition (2.13a) implies that $Q_1 = q_0$ so that the relation (2.9c) is automatically satisfied for $i = 1$ (assuming that the initial values are consistent). The condition (2.13b) implies that the nonlinear system (2.9a,b,c) is independent of $\Lambda_s$. It is possible to fix the parameter $\Lambda_s$ in (2.9e) in such a way that $0 = G(q_1)H_p(q_1, p_1)$ is satisfied. Therefore it holds $\Phi_h(\mathcal{M}) \subset \mathcal{M}$ and the numerical solution stays on the same manifold as the exact solution does.

**The Lobatto IIIA - IIIB Combination.** A pair of Runge-Kutta methods which satisfies all the above properties is the combination of the Lobatto IIIA (in the role of $a_{ij}, b_i$) and Lobatto IIIB (in the role of $\widehat{a}_{ij}, \widehat{b}_i$) methods ([Ja]). For $s = 2$ this is just the combination of the trapezoidal rule with the implicit midpoint rule (the so-called *Rattle* algorithm [An]). For $s = 3$ the coefficients are given below.

Lobatto IIIA (order 4)

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1/2 | 5/24 | 1/3 | − 1/24 |
| 1 | 1/6 | 2/3 | 1/6 |
| | 1/6 | 2/3 | 1/6 |

Lobatto IIIB (order 4)

| | | | |
|---|---|---|---|
| 0 | 1/6 | − 1/6 | 0 |
| 1/2 | 1/6 | 1/3 | 0 |
| 1 | 1/6 | 5/6 | 0 |
| | 1/6 | 2/3 | 1/6 |

Besides the above mentioned properties it can be shown that the nonlinear system (2.9a,b,c) possesses a locally unique solution. It can be solved efficiently by combining fixed point iteration with simplified Newton iteration. The superconvergence (order $2s - 2$) for these methods has been proved by L. Jay. At the moment, a variable step size implementation of these methods is under investigation.

## References

[An] H.C. Anderson (1983): *Rattle: a velocity version of the Shake algorithm for molekular dynamics calcultations*. J. Comput. Phys. **52**, 24-34.

[Ha] E. Hairer (1994): *Backward analysis of numerical integrators and symplectic methods*. Annals of Numerical Mathematics **1**, 107-132.

[HaNW] E. Hairer, S.P. Nørsett & G. Wanner (1993): *Solving Ordinary Differential Equations I. Nonstiff Problems. Second Revised Edition*. Springer Series in Computational Mathematics **8**, Springer-Verlag.

[Ja] L. Jay (1994): *Symplectic partitioned Runge-Kutta methods for constrained Hamiltonian systems*. To appear in SIAM J. Numer. Anal.

[Ja2] L. Jay (1994): *Runge-Kutta-type methods for differential-algebraic equations of index 3 with application to Hamiltonian systems*. Thèse, Section de Mathématiques, Université de Genève, 168 pp.

[LeS] B.J. Leimkuhler & R.D. Skeel (1993): *Symplectic numerical integrators in constrained Hamiltonian systems*. To be published.

[Re] S. Reich (1993): *Symplectic integration of constrained Hamiltonian systems by Runge-Kutta methods*. To be published.

[SaSC] J.M. Sanz-Serna & M.P. Calvo (1994): *Numerical Hamiltonian Problems*. Applied Mathematics and Mathematical Computation **7**, Chapman & Hall.

# High Performance Computing and Networking – Applications –

Wolfgang Joppich
Institute for Algorithms and Scientific Computing
Gesellschaft für Mathematik und Datenverarbeitung mbH
P.O. Box 1316, 53731 Sankt Augustin, Germany

## 1 General Remarks

The systematic use of high performance computers to solve complex problems in science, industry and commerce by mathematical modeling and computersimulation has developed into an independent methodology besides theory and experiment. High performance computing in this context denotes the treatment of problems which pose high demands to the computer resources. Although there exists no exact definition where high performance computing starts, one should think at applications whose solution with adequate response time requires several hundreds of Megaflops combined with large (internal and external) memory.

The platforms for such applications are multi-vectorcomputers, scalable (highly or massively) parallel machines and as a special platform workstation cluster. The commercial availability of such machines opens new perspectives especially to disciplines like molecular chemistry, medicine, drug design, biotechnology, weather forecast, crash simulation, computational fluid dynamics, microelectronics – just to count some application fields.

For many of these applications parallel architectures will be the platform to work with. Apart from the algorithmic side, the central problem of parallel computing is the software. The software problem is the main obstacle that lies in the way of wide-spread industrial use of parallel computers. Overcoming this difficulty and achieving a breakthrough in industrial parallel computing has to be one of the main short-range goals of computational science. This paper shows some examples where the GMD's Institute for Algorithms and Scientific Computing (SCAI) has been working on with exactly this goal.

For most of our parallelized applications, the parallelization strategy is given by *grid partitioning* or, in the absence of grids, by *data partitioning*. Within our work, the following four requirements reflect our goals of working in parallel computing.

1. **Numerical efficiency.** To base parallel developments on numerically efficient algorithms is, in the long run, the only way to obtain the highest gain by parallel computing. Because of their numerical "optimality", multigrid methods play an outstanding role.

2. **Parallel efficiency.** The benefit of numerically efficient (sequential) algorithms can only be maintained if the parallelization introduces as little overhead as possible. In particular, communication requirements have to be minimized.

3. **Scalability.** With increasing number of processing nodes, scalability of an algorithm becomes of major concern. Only numerically highly efficient (optimal) algorithms can be expected to be truly scalable.

4. **Portability.** Clearly, this is a *conditio sine qua non* for the development of large programs unless specific hardware is the target of the work.

## 2 Multigrid Methods for VLSI Process Simulation

The numerical solution of the diffusion equation in VLSI process simulation leads to large systems of nonlinear equations which have to be solved at every time step. Because of the oxidation of silicon the physical domain may vary in time (here: changing from a rectangle to the well known "bird's beak geometry"). For such a situation a multigrid (MG) algorithm with locally refined grids has been constructed.

To guarantee a clear structure of the algorithm which will carry over to diffusion problems in other applications or heat transfer problems there are several algorithmical decisions.

1. Time discretization is done by implicit schemes of first (Backward Euler, BE) and second (Crank-Nicolson, CN) order.

2. Space discretization uses second order schemes on the original domain. A special discretization of the boundary condition on curved boundaries is used.

3. The nonlinear solver for every time step is an MG-algorithm which provides an estimation of the discretization error automatically.

4. The domain of refinement is determined cheaply on relatively coarse grids with the aid of an estimate of the discretization error.

5. An automatic time stepping is performed.

6. Special monotone and shape preserving interpolation schemes are used.

The overall behavior of the algorithm shows that the influence of the nonlinearity is stronger than the movement of the boundary due to the oxidation of silicon. Comparisons of the time discretization methods in combination with the time stepping show the advantage of CN to BE even if the time step size of CN is limited by an estimation of the stability condition of CN for nonlinear problems (Figure 1).

To construct efficient MG-algorithms one has to select the components carefully. Of special importance is the choice of the smoothing operator. Smoothing analysis, a tool for selecting appropriate relaxation methods within an MG-algorithm, shows a good agreement between theoretically expected and experimentally observed MG convergence rates for lexicographic Gauss-Seidel Relaxations. For this application, the MG convergence rate does (in most practical situations) not strongly depend on the time step size, because the smoothing rates of similar linear model problems are upper bounds which are approximated only asymptotically with growing $\Delta t$ (Figure 2).

Of special interest are interpolation techniques which are able to interpolate extremely varying values without oscillations. Different monotone interpolation techniques which do not depend on the difference operator provide non-oscillating grid functions. Because the local order of approximation is sufficiently high these techniques are also used within the full multigrid (FMG) algorithm. These interpolations are important, too, in providing new grid function values on refinement areas which are created from one time step to another.

Locally refined grids are necessary to save computing time. The region of refinement can be controlled by a numerically motivated criterion based upon the local discretization error which is estimated with the aid of the "relative local discretization error of two consecutive grids". This quantity is computed cheaply and automatically within a nonlinear variant of the MG-algorithm (FAS-scheme) on relatively coarse grids. Only a few MG-iterations are necessary to provide a good estimation of the discretization error which guarantees an appropriate size of the refinement area.

For more details concerning multigrid methods and this application see [8].

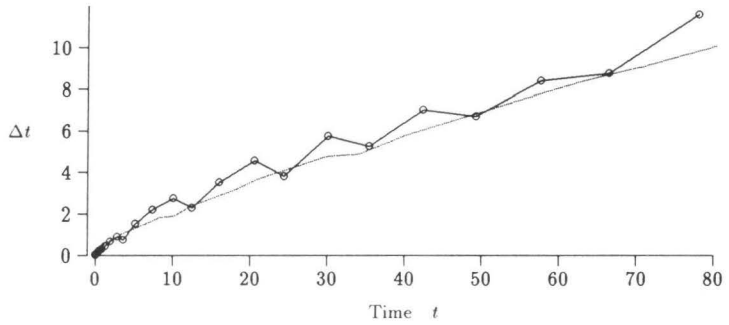



Figure 1: Development of time step size/nonlinear stability condition, $\circ\hat{=}$ MGI-43 CN

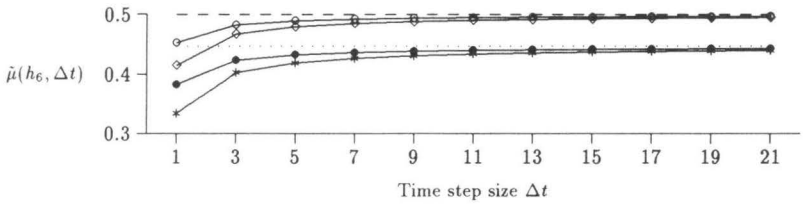



Figure 2: Smoothing rates for growing $\Delta t$ on $\tilde{\Omega}_6$, t=80.0,
$\circ\hat{=}$ PGSN,BE, $\bullet\hat{=}$ SGSN,BE, $\diamond\hat{=}$ PGSN,CN, $*\hat{=}$ SGSN,CN

# 3 Portable Parallelization of the IFS

The IFS ( Integrated Forecasting System) is the European weather prediction model that is run daily by the European weather center ECMWF as a supercomputer production code. It is parallelized in a portable way.

The prognostic variables wind, temperature, humidity and pressure are determined with the aid of initial information concerning orography, vegetation, sun radiation, clouds, ice and snow. The numerical model uses a global grid consisting of more than four millions of points (320 latitudes $\times$ 640 longitudes $\times$ 31 vertical levels) to calculate the unknows per time step (approximately 900 time steps for a ten days period).

The IFS consists of approximately 220,000 lines of Fortran code and supports both a variety of meteorological models and several algorithmic alternatives. Problems of the parallelization are caused by the variety of data structures employed in the code (one triangular and two rectangular data spaces are used by the spectral transform method), which requires non-trivial mappings of the data to the processes and special considerations for the efficient parallelization and the data structuring in the parallel program.

The sophisticated parallelization strategy (data transposition between the grid and the spectral space) gives parallel efficiencies of 80% and more on most relevant parallel systems available today.

## 3.1 Basic Algorithm and Decisions

The solution method employed in the IFS is the spectral transform technique using triangular truncation. In order to exploit the fact that the spherical harmonics are eigenfunctions of an essential part of the underlying operator, some parts of the calculations are performed in spectral space. Other

computations such as the whole physics are carried out in the grid point space. Altogether, three different discrete function spaces are involved in the calculations: grid point space, Fourier space and spectral space. The corresponding discrete functions are the spectral coefficients, the Fourier coefficients and the Gaussian grid point values.

A detailed description of the spectral transform technique can be found, for example, in [5, 12]. The calculations in the different algorithmic steps involve complex data dependencies in different directions. For example, data dependencies with respect to the zonal wave numbers and such with respect to the latitudes and such with respect to the longitudes. The physical computations involve vertical data couplings.

The parallelization of such a big code as the IFS with a lifetime of about 10 years or more requires *portability* among all present and future parallel machines of interest. Nobody wants to parallelize such a code again and again for each new parallel machine. Therefore, the portable PARMACS message passing interface developed at the GMD was selected for the parallelization of the IFS, with a possible switch to MPI, as soon as it will be available. This is in accordance with similar decisions in several current European projects.

A careful analysis showed that the data transposition strategy [1] was a promising approach. The idea of this strategy is to re-distribute the complete data to the processes at various stages of the algorithm such that the arithmetic computations between two consecutive transpositions can be performed without any interprocess communication.

### 3.2 The Data Transposition Strategy and the Partitioning of Data

The data transposition approach comprises advantageous features with respect to the parallel efficiency as compared with alternative parallelization approaches. Moreover, it allows to keep the main numerical components of the sequential algorithm unchanged.

The transposition strategy can be detailed as follows: In grid point space, the data partitioning is over latitudes, whereas during the calculations in spectral space and in Fourier space, the data are distributed with respect to the zonal wave numbers. The switch between these two different partitionings is performed in Fourier space. This means that the Fourier coefficients are re-distributed twice in each time step. This approach has the obvious advantage that each of the single algorithmic steps can be performed without any interprocess communication.

For parallel systems with up to several hundreds of processors a 1D partitioning of the data in each of the different data spaces is adequate. In grid point space and for the direct and inverse Fourier transforms, the partitioning is over latitudes, i.e. different processes are responsible for the calculations of different (whole) latitudes of the underlying regular Gaussian grid, whereas in spectral space and for the direct and inverse Legendre transforms, the data are distributed with respect to the zonal wave numbers. The data distributions in these two data spaces determine automatically the data partitionings in the Fourier space, where the transpositions are performed.

Load balancing can easily be taken into account in grid point and Fourier space since the corresponding data are described by 2D or 3D Fortran arrays, *provided that the number of latitudes is a multiple of the number of processes to be utilized for the parallel application.* If this condition is violated, some performance loss has to be expected. The partitioning over latitudes is performed in a way that each processor is responsible for a group of adjacent latitudes.

Due to the triangularity of the data structures in spectral space, load balancing issues have to be taken into account explicitly in such data distributions. It should be noted that some load imbalance in the spectral space computations is of minor importance for the full 3D model since, in this case, the amount of work spent in spectral space is almost negligible as compared with the total work.

### 3.3 Performance Results

The parallelized 2D (essentially using only one level) and 3D codes have been run successfully on various parallel systems. Of course, the 3D code has much higher demands with respect to the system resources than the IFS-2D. Compared with the 2D model, the characteristic feature of the corresponding 3D version is the inclusion of the physics computations, which correspond to the consideration of a multitude of meteorological models. About half of the whole computing time is spent in these meteorological subroutines [5].

In this section, we present results for resolutions ranging from T10L19 to T106L19, where T$m$L$k$ denotes the truncation number $m$ for the spectral space computations, and $k$ the number of vertical levels employed. In the physical space, this corresponds to resolutions ranging from 32 longitudes $\times$ 16 latitudes to 320 longitudes $\times$ 160 latitudes.

Tables 1 and 2 show the performance of the IFS 3D on the CM5 and the SP1 at the GMD. The results are excellent; in all cases, the parallel efficiencies on the CM5 and the SP1 are higher than 90 % and 80 %, respectively. It should be noted that vectorization was not yet utilized for the CM5 results.

Table 1: Speed–up, parallel efficiency and computing times on CM5; 10 time steps, compiler option: -O; no vectorization.

| Processors | | 1 | 2 | 4 | 8 | 11 | 16 | 22 | 32 |
|---|---|---|---|---|---|---|---|---|---|
| Efficiency (%) | T10L19 | 100 | 99 | 98 | 96 | – | – | – | – |
| | T21L19 | – | 99 | 97 | 96 | 90 | 96 | – | – |
| | T42L19 | – | – | – | 97 | 93 | 96 | 92 | 93 |
| Time (sec) | T10L19 | 249 | 126 | 63 | 33 | – | – | – | – |
| | T21L19 | – | 494 | 254 | 128 | 98 | 65 | – | – |
| | T42L19 | – | – | – | 526 | 396 | 265 | 203 | 137 |

Table 2: Parallel efficiencies and computing times in seconds on the IBM SP1 (compiler option: -O) for the portable parallel IFS-3D code; ten time steps.

| Processors | | 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|
| Efficiency (%) | T42L19 | 100 | 94 | 90 | 81 |
| | T63L19 | – | 94 | 89 | 84 |
| | T106L19 | – | – | 91 | 87 |
| Time (sec) | T42L19 | 588.5 | 316.4 | 163.5 | 105.9 |
| | T63L19 | – | 729.4 | 393.2 | 142.0 |
| | T106L19 | – | – | 1218.3 | 636.7 |

One striking feature of the parallel performance of the 3D code is that the parallel efficiencies are even higher than in the 2D case [6]. The reason for this is that the computational work increases with the number of levels *and* with the physics computations, while the latter does not influence the communication costs. Therefore, the ratio of arithmetic and communication is much better for the parallelization when dealing with the 3D case. Moreover, as already mentioned before, load imbalances in spectral space are of minor importance in the 3D case.

The memory requirements of the 3D-code are severe and grow in a nonlinear manner. Additionally, a large number of fields in the physics part grow with increasing problem size, independent of the number of processes utilized. Thus, only resolutions of up to T42L19 could be run on the

CM5 of GMD (32 Mbyte per node). The eight nodes of the SP1 at the GMD are equipped with 128 MBytes of memory. T106L19 (320 longitudes, 160 latitudes, 19 levels) is the largest problem which can be run on this system. This resolution was the previous operational model of the ECMWF; today T213L31 ($640 \times 320 \times 31$) is standard.

The parallel program has been run by the ECMWF's Parallel Computing group on the Cray C90 at Reading. The following results have been achieved running on a "quiet" C90 using the PALLAS implementation of PARMACS for Cray C90.

Table 3: Speed-up and computing times on Cray C90

| Processors | | 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|
| Speed-up | T42L19 | 1 | 1.96 | 3.6 | 6.8 |
| | T63L19 | 1 | 1.98 | 3.8 | 7.0 |
| | T106L19 | 1 | 1.97 | 3.8 | 7.2 |
| Time (sec) | T42L19 | 29.3 | 14.9 | 8.1 | 4.3 |
| | T63L19 | 67.9 | 34.3 | 17.8 | 9.7 |
| | T106L19 | 184.3 | 93.3 | 48.0 | 25.7 |

Table 3 exploits the vector merge function inlining which allows all loops to vectorize as normal on Cray vector computers.

# 4 Parallel multigrid for CFD

In the POPINDA project, a cooperation between DLR, Dornier, DASA, Deutsche Airbus, IBM and the GMD, 3 dimensional parallel compressible Navier-Stokes solvers, based on a 3 dimensional communications library will be developed. In the first stage the suitability of using parallelization strategies of already existing two dimensional codes for the 3D solvers is investigated. At GMD a two dimensional Euler code for block-structured grids has been developed as part of the LiSS package [11], which is a program package to solve partial differential equations on general two-dimensional domains. From the beginning of the package development the parallel solution of equations has been focussed upon by constructing efficient parallel multigrid methods and parallelization tools, like a two-dimensional communications library, based on PARMACS ([3]). The parallelization strategy adopted is the grid partitioning method (explained in [13]). In grid partitioning the grid is split into blocks, which are mapped to different processes. The quality of the parallelized multigrid algorithm is investigated here for compressible Euler equations. A flow problem around an airfoil is taken and the domain is split into many blocks (up to 256). When a grid is split into many blocks, which are all smoothed simultaneously multigrid looses its h-independent convergence. In ([10]) this was observed and treatments to overcome this problem were given for the incompressible Navier-Stokes equations. These treatments are applied here. Furthermore, the multigrid code is tested on several parallel machines, like the IBM SP1 with 10 nodes, the CM5 with 64 nodes and a cluster of workstations containing 17 nodes.

## 4.1 Parallel multigrid

The parallel multigrid algorithm consists of a host and a node program. The host program takes care for the organization of in- and output, creates node processes, mails initial data to node processes and receives calculated results, like residuals. These tasks are taken care for by PARMACS-based ([3]) routines of the 2-D Communications Library. In the node program the calculation takes place, also the communication between nodes is taken care for. Grid partitioning, the technique to distribute parts of a domain to different processes, is explained for example in [13]. Along the interior block

boundaries, an overlap region is placed, and all operations in multigrid, restriction, prolongation and smoothing can be performed in parallel. Keeping values in overlap regions up-to-date requires communication between nodes. A first order discretization requires an overlap of one line of cells in order to achieve accuracy, while for the second order the stencil for evaluation of the right-hand-side grows and an overlap of two cells is needed for accuracy. In [10] with 32 blocks a similar convergence factor was obtained as for single block with an extra internal boundary relaxation after each smoothing sweep and with an extra update of the overlap region, when smoothing along lines in the first direction of the alternating method was finished. This strategy is investigated here for 2D Euler equations with Coupled Damped Alternating Line-Gauss-Seidel (DALGS) per block as smoother in a multigrid F-cycle. A problem exists in grid coarsening. At a certain stage on a coarse level every process contains the minimal number of grid points. However, when the domain is split into many blocks this coarse level does not need to be the global coarsest grid. With an agglomeration strategy ([7]) grids can be coarsened further leaving some processors idle and re-mapping the busy processors to an optimal configuration with little communication time. Here an agglomeration strategy is not yet implemented; all processors contain a minimal number of grid points per block on the coarsest grid level. When a $257 \times 65$-grid is split into 256 equal sized $9 \times 9$ blocks three multigrid levels remain with a one cell overlap.

## 4.2 Results

The computational domain around a NACA0012 airfoil is a C-grid consisting of $257 \times 65$ cells. The popular transonic testcase presented here is: Mach number $M_\infty = 0.85$ and angle of attack $\alpha = 1^o$ with a strong lee-side shock and a less strong wind-side shock (see Figure 3).

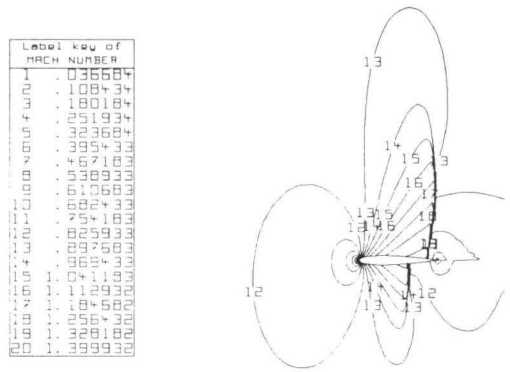



Figure 3: Mach number distribution for flow around an airfoil, $M_\infty = 0.85, \alpha = 1^o$.

First, for all splittings the influence of an additional update of the overlap region and/or an extra interior boundary relaxation on the convergence factor is investigated for the first order discretization. The details of the nonlinear multigrid algorithm are: FAS F-cycle with nested iteration and 1 pre- and 1 post-smoothing (=F(1,1)) is used. Underrelaxation $\omega$ for DALGS depended on the number of blocks into which the grid was partitioned. This is probably due to the absence of an agglomeration strategy. For 1 to 16 block splittings $\omega = 0.8$ resulted in satisfactory convergence, for 64 blocks $\omega = 0.7$; for 256 blocks $\omega = 0.45$ was best. It can be seen that with one additional

boundary relaxation every smoothing step on interior block boundaries and with an extra update of overlap the multigrid convergence is not much affected by grid partitioning.

Also the performance of the algorithm is investigated on a cluster of 17 workstations, on IBM SP1 with 10 nodes and on CM5 Connection Machine with 64 nodes. Due to the implementation consisting of a host and a node program, it is necessary to use one node as host. Therefore, the maximal number of blocks is taken to be 32 on the CM5. For all splittings the same algorithm is investigated: 5 F(1,1) cycles with two additions, started with nested iteration are performed. Figure 4 shows solution times for several splittings on different computers.

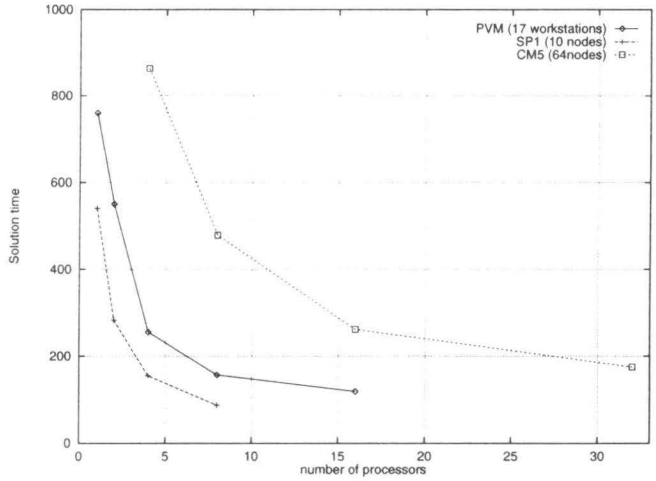



Figure 4: *Solution times for different partitionings on different machines.*


## Acknowledgements:

The presented work and the achieved results have been provided by project teams of the Institute for Algorithms and Scientific Computing at the GMD: U. Gärtel, W. Joppich and A. Schüller produced the contribution concerning the IFS while K. Oosterlee, H. Ritzdorf, A. Schüller and B. Steckel worked on the parallel CFD application with the PDE-solver $L_iSS$.


## References


[1] S.R.M. Barros and T. Kauranne, *Spectral and multigrid spherical Helmholtz equation solvers on distributed memory parallel computers.* Proceedings of Fourth Workshop on Use of Parallel Processors in Meteorology, 26–30 November 1990, ECMWF, 1–27.

[2] R. Calkin, R. Hempel, H.-C. Hoppe and P. Wypior, *Portable programming with the PARMACS message passing library.* Parallel Computing **20** (1994) 615–632, North–Holland.

[3] L. Bomans, R. Hempel, D. Roose, *The Argonne/GMD macros in FORTRAN for portable parallel programming and their implementation on the Intel iPSC/2.* Parallel Computing **15** (1990) 119–132, North–Holland.

[4] J. Canu, J. Linden, *Multigrid solution of 2D Euler equations: A comparison of Osher's and Dick's flux difference splitting schemes.* GMD Arbeitspapiere 693, GMD St. Augustin, Germany (1992).

[5] D. Dent and A. Simmons, *The ECMWF multi-tasking weather prediction model.* Computer Physics Reports **11** (1989) 153–194, North-Holland.

[6] U. Gärtel, W. Joppich and A. Schüller, *Portable parallelization of the ECMWF's weather forecast program. Technical documentation and results.* Arbeitspapiere der GMD **820**, Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin, 1994.

[7] R. Hempel, A. Schüller, *Experiments with parallel multigrid algorithms using the SUPRENUM communications subroutine library.* GMD Arbeitspapiere 141, GMD St. Augustin, Germany (1988).

[8] W. Joppich and S. Mijalković. *Multigrid Methods for Process Simulation.* Computational Microelectronics. Springer Verlag Wien, New York, 1993.

[9] B. Koren, Defect correction and multigrid for an efficient and accurate computation of airfoil flows. *J. Comp. Phys.* **77**, 183–206 (1988).

[10] G. Lonsdale, A. Schüller, Multigrid efficiency for complex flow simulations on distributed memory machines. *Parallel Comp.* **19**, 23–32 (1993).

[11] G. Lonsdale, H. Ritzdorf, K. Stüben, *The LiSS package.* GMD Arbeitspapier 745, GMD St. Augustin, Germany (1993)

[12] B. Machenhauer, *The spectral method.* Numerical Methods Used in Atmospheric Models, GARP publication series No. **17** (1979) 121–275.

[13] O.A. McBryan, P.O. Frederickson, J. Linden, A. Schüller, K. Solchenbach, K. Stüben, C.A. Thole, U. Trottenberg, Multigrid methods on parallel computers - a survey of recent developments. *Impact Comp. Science and Eng.* **3**, 1–75 (1991).

[14] S.P. Spekreijse, *Multigrid solution of the steady Euler equations*, CWI Tract 46 (1988).

# *Extended Abstract for the 1994 Dutch Numerical Analysis Seminar:* Newton-Krylov-Schwarz Domain Decomposition Methods for Partial Differential Equations in Science and Engineering

David E. Keyes *

## 1 The Implicit Imperative

Few production codes for the nonlinear elliptic boundary value problems that arise in science and engineering as the expression of fundamental conservation laws are fully implicit. A noteworthy exception is the full potential TRANAIR code [15], which accounts for an enormous number of cycles in Boeing's Commercial Airplane Division. (Our discussion relies on computational fluid dynamics for illustration, but most of it is applicable throughout computational mechanics.) Implicit methods will be increasingly important in applications because of the desire to perform tens or hundreds of analyses in conjunction with design optimization. Analysis methods capable of reliably and rapidly producing low-residual solutions seem prerequisite to practical multidisciplinary optimization. Newton-Krylov methods are well suited for the implicit solution of nonlinear problems whenever it is unreasonable to compute or store a true Jacobian. Krylov-Schwarz or "domain decomposition" iterative methods are well suited for the parallel implicit solution of multidimensional systems of boundary value problems, and have been undergoing active development for the past seven years [4, 5, 6, 8, 9, 12, 13, 14] They provide good data locality so that even a high-latency workstation network can be employed as a parallel machine. We call the combination of these two methods Newton-Krylov-Schwarz (NKS) and have been investigating some of their algorithmic and implementation aspects. Though they

*Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, and Institute for Computer Applications in Science and Engineering (ICASE), MS 132C, NASA Langley Research Center, Hampton, VA 23681-0001, keyes@icase.edu. The work abstracted herein was supported by NSF grant ECS 89-57475, by NASA contract No. NAS1-19480, and by the Boeing Company, the Ford Motor Company, and the United Technologies Corporation. Portions were adapted from the introduction of *Domain-based Parallelism and Problem Decomposition Methods in Science and Engineering*, D. E. Keyes, Y. Saad, and D. G. Truhlar, eds., SIAM, Philadelphia (to be published, 1995).

are still in their infancy, we believe that they will increasingly occupy the creative attention of developers thoughout computational mechanics. Indeed, until the goals of Newton-Krylov-Schwarz methods are routinely delivered in analysis codes, some "computational grand challenge" problems will need to await.

## 2 Newton-Krylov Methods

Newton-like methods in principle allow a more rapid asymptotic approach to steady states than do time-explicit methods or semi-implicit methods based on defect correction. Strict Newton methods have the disadvantage of requiring solutions of linear systems of equations based on the true Jacobian of the steady nonlinear residual and are often impractical in several respects: (1) their quadratic convergence properties are realized only asymptotically; (2) some popular discretizations (e.g., using limiters) of the residual are nondifferentiable; (3) the Jacobian is often inconvenient or expensive to form and store; (4) the Jacobian may have a bad condition number; and (5) the most popular family of preconditioners, incomplete factorization, is difficult to parallelize efficiently. The first two disadvantages need to be addressed in problem-specific ways; sequencing the mathematical models of the phenomenon under investigation and the grid resolution of a given model can usually steer a computation into the domain of convergence; see, e.g. [7]. We have been concentrating on the last three of these points, the most important with respect to parallel computational fluid dynamics.

High-accuracy evaluation of the discrete residuals of multidimensional flow requires a large number of arithmetic operations. Their Jacobians, though block-sparse, have dense blocks and are usually an order of magnitude even more complex to evaluate. Hence, matrix-free Newton-Krylov methods, in which the action of the Jacobian is required only on a set of given vectors, instead of on a full basis, are natural in this context. To solve the nonlinear system $f(u) = 0$, given $u^0$, we let $u^{l+1} = u^l + \lambda^l \delta u^l$, for $l = 0, 1, \ldots$, until the residual is sufficiently small, where $\delta u^l$ approximately solves the Newton correction equation $J(u^l)\delta u^l = -f(u^l)$, and parameter $\lambda^l$ is selected by some line search or trust region algorithm. Krylov methods, such as the method of conjugate gradients for symmetric positive definite systems or GMRES for general nonsingular systems, find the best approximation of each Newton correction in a relatively small-dimensional subspace that is built up from successive powers of the Jacobian on the current nonlinear residual.

The action of Jacobian $J$ on an arbitrary Krylov vector $w$ can be approximated by $J(u^l)w \approx \frac{1}{\epsilon}\left[f(u^l + \epsilon w) - f(u^l)\right]$. Finite-differencing with $\epsilon$ makes such matrix-free methods potentially much more susceptible to finite word-length effects than ordinary Krylov methods, and the bad condition numbers that are typical of multiple-scales problems may lead to unacceptably large iteration counts. An approximation to the Jacobian can be used to precondition the Krylov process. Examples are: (1) the Jacobian of a lower-order discretization; (2) the Jacobian of a related discretization that allows economical analytical evaluation of elements; (3) a finite-differenced Jacobian computed with lagged values for ex-

pensive terms; and (4) domain decomposition parallel preconditioners composed of Jacobian blocks on subdomains of the full problem domain. Case (4) can be combined with any of the other split-discretization techniques, in principle, and introduces the additional appellation of "Schwarz".

## 3 Krylov-Schwarz Methods

"Think globally; act locally." This bumper sticker maxim has a lot to say to practitioners of contemporary high-performance computing. With the pressure on memory size unrelenting, it is incumbent on all users to respect the data access hierarchies that accompany large memories. These hierarchies are imposed, ultimately, by the finite speed of light but their presence is asserted more immediately by the software overhead of system protocols for the delivery of data. From the frame of reference of any given processing element, a cost function can be constructed that gives the minimum time required to access a memory element any given distance away. Such functions are typically plateaus separated by sharp discontinuities that correspond to software latencies when some boundary of the hierarchy, such as a cache size or a local memory size, is crossed. The locations of these discontinuities should expressly be respected by user applications.

Fortunately, many problems are adaptable to such memory hierarchies. Steady-state natural and human-engineered systems are often zero-sum networks in which the overall distribution of a quantity to be determined is conserved. The conservation principle holds over any size control volume, from the smallest scales requiring resolution up to the global domain. Somewhere between these extremes are the scales at which the latencies of the memory hierarchy are asserted. This suggests a multilevel discretization of the conservation laws, with coarse-grained interactions between "basins" of fast memory, and fine-grained interactions within them [11]. Algorithms exploiting multilevel discretization have evolved naturally and somewhat independently in a variety of applications, both continuous and discrete.

It may be assumed without loss of generality that the challenges of writing algorithms for large-scale problems on hierarchical memory systems occur for physical systems that are irreducible in the matrix theoretic sense of the term. Each degree of freedom depends upon all of the others; no degrees of freedom may be removed and solved for exactly in isolation. For irreducibly coupled physical systems with arbitrary interactions between the components, it is not obvious that a decomposition of the unknowns of the problem into sets that are physically proximate, and a mapping into the global memory in a way that preserves their proximity, will have any benefit. However, interactions in diffusive systems decay with distance sufficiently rapidly that remote interactions may be "lumped" in certain phases of the solution process.

A variety of parallel preconditioners, whose inverse action we denote by $B^{-1}$, can be induced by decomposing the domain of the underlying PDE, finding an approximate representation of $J$ on each subdomain, inverting locally, and com-

bining the results. Generically, we seek to approximate the inverse of $J$ by a sum of local inverses, $B^{-1} = R_0^T J_{0,u^l}^{-1} R_0 + \sum_{k=1}^{K} R_k^T J_{k,u^l}^{-1} R_k$ , where $J_{k,u^l} = \left\{ \frac{\partial f_i(u^l)}{\partial u_j} \right\}$ is the Jacobian of $f(u)$ for $i$ and $j$ in subdomain $k$ ($k > 0$), subscript "0" corresponds to a possible coarse grid, and $R_k$ is a restriction operator that takes vectors spanning the entire space into the smaller-dimensional subspace in which $J_k$ is defined.

The simplest of these Schwarz preconditioners is block Jacobi, which is a zero-overlap form of additive Schwarz. The convergence rate of block Jacobi can be improved, at higher cost per iteration, with subdomain overlap and (for many problems) by solving an additional judiciously chosen coarse grid system. It has been proved theoretically for the selfadjoint case and illustrated numerically for a variety of nonselfadjoint scalar elliptic problems that additive Schwarz with a nested coarse grid, containing one degree of freedom per subdomain, provides an "optimal" preconditioning, in the sense that the number of iterations required to attain a fixed reduction in residual is bounded by a constant as either the mesh spacing $h$ or the diameter of the subdomains $H$ is indefinitely refined [2]. Multiplicative Schwarz methods improve on additive methods, like block Gauss-Seidel improves upon block Jacobi, but impose a serialization penalty. In a situation in which there are more subdomains than processors, hybrid multicolored multiplicative/additive Schwarz is recommended for optimal convergence at a given parallel granularity.

Parallelism is not the sole motivation for Schwarz methods. We remark that, given a preconditioner for the global domain, a Krylov-Schwarz method in which the same preconditioner is applied locally on each subdomain may provide a better serial algorithm than Krylov acceleration of the original global preconditioner. Given a problem of size $N$ and a preconditioner with arithmetic complexity $c \cdot N^\alpha$, partition the problem into $P$ subproblems of size $N/P$. The complexity of applying the solver independently to the set of subproblems is $P \cdot c \cdot (N/P)^\alpha$. Even in serial, $P^{\alpha-1}$ sets of subdomain iterations can be afforded to coordinate the solutions of the subproblems per single global iteration, while breaking even in total complexity. If $\alpha > 1$, there is "headroom" for the domain-decomposed approach, depending upon the overall spectral properties of the global and multidomain preconditioners. There may still be parallel headroom even if $\alpha = 1$, since the global method may involve too much communication to parallelize efficiently. In addition, a hierarchical data structure is often natural for modeling or implementation reasons; and memory requirements, cache thrashing, or I/O costs on large problems may demand decomposition anyway.

## 4 Newton-Krylov-Schwarz Methods

Newton-Krylov-Schwarz is a natural outcome of the imperatives of implicitness and parallelism for nonlinear PDEs and results from the domain decomposition of any global preconditioner for Newton-Krylov. NKS techniques are similar to nonlinear defect correction schemes in that a low-order operator ($J_L$) is employed

on the left-hand side in order to drive a high-order residual on the right-hand side ($f_R$) to zero. The difference lies in the implicit use of a high-order Jacobian ($J_R$) on the *left-hand* side of a NKS method. The approximate Jacobian, $J_L$, is employed on a subdomain scale and used to precondition the implicitly defined true Jacobian, $J_R$, at each implicit time step. In matrix terms, defect correction solves $J_L \delta u = -f_R$, and NKS solves $[\sum_{k=1}^{K} R_k^T J_{L,k}^{-1} R_k] J_R \delta u = -[\sum_{k=1}^{K} R_k^T J_{L,k}^{-1} R_k] f_R$. The Schwarzian convergence theory relies on the dominance of the elliptic operator. Multidimensional hyperbolic problems are often discretized with an artificially diffusive left-hand side operator, however. Partial benefit of the elliptic algorithms will be realized when these methods are accelerated with a Krylov method.

Several aspects of Newton-Krylov-Schwarz methods that are expected to be of value in a practical parallelized Navier-Stokes code have been validated in model contexts (convection-diffusion, potential, Euler, Navier-Stokes) [1, 3, 10]. Simultaneous demands for higher-order discretizations in the Jacobian and easily invertible preconditioners are satisfied by mixed (inconsistent) discretizations. The use of a nonnested coarse grid in the preconditioner provides a continuously adjustable "knob" with significant leverage in reducing the condition number of an elliptically dominated problem. The coarse grid entails a high communication cost in parallel implementations, but in additive algorithms it can be solved simultaneously with the subdomain fine grid operators, permitting overlap of its excess communication with useful subdomain computation. Finally, preliminary experience with treating Unix workstations connected only by ethernet is encouraging in that even for a modest fixed-size problem, wall-clock execution time per iteration improves on up to sixteen workstations.

## References

[1] J. G. Chefter, C. K. Chu, and D. E. Keyes, *Domain Decomposition for the Shallow Water Equations*, in D. E. Keyes and J. Xu, eds., *Proc. of the Seventh Intl. Symp. on Domain Decomposition Methods in Science and Engineering*, AMS, Providence, 1995 (to appear).

[2] X.- C. Cai, W. D. Gropp, and D. E. Keyes, *A Comparison of Some Domain Decomposition and ILU Preconditioned Iterative Methods for Nonsymmetric Elliptic Problems*, J. Numer. Lin. Alg. Applics., 1994 (to appear).

[3] X.- C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton-Krylov-Schwarz Methods in CFD*, Proceedings of an International Workshop on the Navier-Stokes Equations, (IBM Center, Heidelberg, October 1993) F. Hebeker and R. Rannacher, eds.. Vieweg Verlag, Braunschweig, pp. 17–30, 1994.

[4] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., *Proc. of the Second Intl. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.

[5] T. F. Chan, R. Glowinski, J. Periaux, and O. B. Widlund, eds., *Proc. of the Third Intl. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1990.

[6] T. F. Chan and T. P. Mathew, *Domain Decomposition Algorithms*, Acta Numerica, pp. 1–59, 1994.

[7] A. Ern, V. Giovangigli, D. E. Keyes and M. D. Smooke, *Towards Polyalgorithmic Linear System Solvers for Nonlinear Elliptic Problems*, SIAM J. Sci. Comp. **15**:681–703, 1994.

[8] R. Glowinski, G. H. Golub, G. A. Meurant, and J. Periaux, eds., *Proc. of the First Intl. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988.

[9] R. Glowinski, Yu. A. Kuznetsov, G. A. Meurant, and O. B. Widlund, eds., *Proc. of the Fourth Intl. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1991.

[10] W. D. Gropp, D. E. Keyes, and J. S. Mounts, *Implicit Domain Decomposition Algorithms for Steady, Compressible Aerodynamics*, Proceedings of the Sixth International Symposium on Domain Decomposition Methods, (Como, June 1992) A. Quarteroni et al., eds., Amer. Math. Soc., Providence, pp. 203–213, 1994.

[11] D. E. Keyes, *Domain Decomposition: A Bridge Between Nature and Parallel Computers*, in Adaptive, Multilevel and Hierarchical Computational Strategies, A. K. Noor, ed., Amer. Soc. Mech. Eng., New York, pp. 293–334, 1992.

[12] D. E. Keyes, T. F. Chan, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., *Proc. of the Fifth Intl. Symp. on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1992.

[13] D. E. Keyes and Jinchao Xu, eds., *Proc. of the Seventh Intl. Symp. on Domain Decomposition Methods in Science and Engineering*, AMS, Providence, 1995 (to appear).

[14] A. Quarteroni, J. Periaux, Yu. A. Kuznetsov, and O. B. Widlund, eds., *Proc. of the Sixth Intl. Symp. on Domain Decomposition Methods in Science and Engineering*, AMS, Providence, 1994.

[15] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti, *A Locally Refined Rectangular Grid Finite Element Method: Application to Computational Fluid Dynamics and Computational Physics*, J. Comp. Phys. **92**:1–66, 1991.

# On the Scalability of the AMLI method

Owe Axelsson
Maya Neytcheva

**Abstract**

The presence of high performance massively parallel computer systems releases the mathematical models from certain restrictions caused by a lack of computer resources such as, for example, processor power available and memory. At first glance, the wish to consider numerically more and more refined and detailed models seems to be reachable just having enough number of processors. But, as a rule, the complexity growth of the model is (still) much faster than the growth of the computing power. Thus, the demand on having efficient solution algorithms is nowadays even more pronounced than in the era of the serial computers. As the experience shows, the existence of optimal and scalable algorithms or, respectively, the absence of these makes meaningful or meaningless the usage of a massively parallel computer.

This contribution addresses numerical algorithms for elliptic partial differential equations which are scalable in the sense that they have optimal computational complexity on serial computers and asymptotically maximal speedup on massively parallel distributed memory computer architectures for a proper balance between the problem size and the number of processors to be used.

As an example of such algorithms, a short version of the Algebraic Multilevel Iteration (AMLI) method is considered. Some numerical illustrations on CM-200 (8K) and CM-2 (64K) are included.

# Diagonally Implicit Iteration Methods for IVP Solvers

**Jacques de Swart (CWI)**

Implicit Runge-Kutta methods for solving the stiff initial-value problem $\{y' = f(y), y(t_0) = y_0\}$ require the solution of systems of non-linear equations. Usually these are dealt with a modified Newton process, in which we have to solve a linear system with coefficient matrix $I - A \otimes J$, where $A$ is the Runge-Kutta parameter matrix and $J$ is an approximation to the Jacobian of $f$. A further modification replaces $A$ by a matrix $B$. In a first investigation at CWI the matrix $B$ was chosen diagonally in order to decouple the linear system in $s$ subsystems ($s$ denotes the number of stages of the Runge-Kutta method) and exploit parallelism across the stages. However, this approach may lead to a slow convergence behavior. In some cases, the residual may even grow before tending to zero. This is a very unpleasant situation if we want to apply so-called parallelism across the steps. In this approach iterates are computed concurrently at a number of time points: as soon as the current iterate at $t_n$, although not yet converged, is sufficiently 'safe' to start iteration at $t_{n+1}$, the method starts iterating at $t_n$ and $t_{n+1}$ in parallel, etc. Of course, in order to get a satisfactory degree of parallelism, we should have fast convergence right from the beginning.

In this talk we will see how applying similarity transformations, as introduced by Butcher in [1], and preconditioning techniques, developed at CWI, leads to an iteration process that has a positive convergence speed right from the beginning. This can be done without loosing parallelism across the stages. Several strategies will be described. This research will be published in [2].

## Bibliography

[1] J.C. Butcher (1976): On the implementation of implicit Runge-Kutta methods, BIT 16, pp. 237–240

[2] P.J. van der Houwen & J.J.B. de Swart (1994): Diagonally Implicit Iteration Methods for IVP Solvers, to appear.

# Optimized Parallel Block Preconditioners Based on Domain Decomposition

Kian H. Tan*and Mart Borsboom†

aug 20, 1994

The continuing demand in CFD for High Performance Computer Models (HPCM) can be satisfied very well by the use of domain decomposition methods (DD-methods). DD-methods facilitate the design of efficient parallel algorithms: they form a class of iterative solution methods where in each iteration step a collection of artificially decoupled and virtually independent subproblems is solved.

Algebraically, DD-methods for a system $Au = f$ can be described in terms of index sets and submatrices of $A$. The partitioning of the vector $u$, or domain decomposition, is determined by a number of index sets, each index set corresponding to a principal submatrix of $A$. Block diagonal preconditioners, with blocks corresponding to the principal submatrices, then give rise to domain decomposition methods which can be parallelized easily.

In our approach we construct block diagonal preconditioners for *extended* systems of equations $\bar{A}\bar{u} = \bar{f}$, with the condition that the original solution $u$ is a subvector of $\bar{u}$. The essential part of this approach is that we have some freedom in the choice of the additional equations. Representation of this freedom by a suitable parameterization of the matrix-enhancement leads to additional equations which can be interpreted as (parameterized) interface conditions. Besides a Dirichlet and a Neumann part, tangential and mixed derivatives are involved in these interface conditions.

For systems originating from discretizations on structured grids, the convergence properties of the block preconditioners can be analyzed and optimized, as a function of the introduced parameters. We present a number of two-dimensional test problems, decomposed in up to 16x16 subdomains, to illustrate the benefits of such optimized block preconditioners for extended systems of equations.

**Key words.** domain decomposition, parallel computing.

*Mathematical Institute, Utrecht University, P.O. box 80010, 3508 TA Utrecht, the Netherlands, E-mail: tan@math.ruu.nl. The work of this author was funded by a grant from Delft Hydraulics, the Netherlands.

†Delft Hydraulics, dept. S&O, P.O. box 152, 8300 AD Emmeloord, the Netherlands, E-mail: Mart.Borsboom@wldelft.nl.

# Recent Developments in Waveform Relaxation Methods

Stefan Vandewalle
Caltech, Applied Mathematics 217-50, Pasadena, CA 91125

**1. Introduction.** Waveform relaxation methods are continuous-in-time iterative methods for solving large systems of ordinary differential equations (ODEs). They have proven to be very successful for solving the systems of ODEs that arise in the simulation of electronic circuits and power systems, and for solving the systems of ODEs derived by semi-discretization of parabolic partial differential equations. Their effectiveness is due to ease with which they can be implemented on multiprocessors, and due to the fact that they allow multirate integration.

**2. Convergence of waveform relaxation.** In the first part of the talk I will recall the waveform relaxation method and outline the theoretical framework used to study its convergence. Consider the linear system of ODEs

$$\dot{U} + AU = F \text{ with } U(0) = U_0 . \tag{1}$$

The method is based on a splitting of $A$. With $A = M - N$, the iteration becomes

$$\dot{U}^{(\nu)} + MU^{(\nu)} = NU^{(\nu-1)} + F \text{ with } U^{(\nu)} = U_0 . \tag{2}$$

This can be rewritten as $U^{(\nu)} = \mathcal{K}U^{(\nu-1)} + \varphi$, with $\mathcal{K}$ a Volterra convolution operator:

$$\mathcal{K}U(t) = k(t) \star U(t) = \int_0^t k(t-s)\,U(s)\,ds \text{ with } k(t) = e^{-Mt}N . \tag{3}$$

The convergence of the iteration is determined by the spectral radius of operator $\mathcal{K}$. The central waveform convergence theorem, due to Miekkala and Nevanlinna ([3]), relates the spectral radius of $\mathcal{K}$ to the spectral radius of a complex matrix:

$$\rho(\mathcal{K}) = \max_{z \in \Sigma} \rho((zI + M)^{-1}N) , \tag{4}$$

where $\Sigma$ is some subset of the complex plane. This result reduces the analysis of the continuous-in-time waveform iteration into a (non-trivial) linear algebra problem.

**3. Acceleration of waveform relaxation.** In the second part of the talk I will survey three recent developments that aim at accelerating the basic waveform iteration.

**Convolution SOR.** Successive overrelaxation (SOR) of the waveform relaxation method has been defined in two ways. Either the splitting of A is chosen in the standard SOR manner, i.e. $M = 1/\omega\, D + L$ and $N = (1-\omega)/\omega\, D - U$ where $D, L$, and $U$ are the diagonal, lower and upper triangular parts of $A$. Or, the splitting is chosen in the Gauss-Seidel way, $M = D + L$ and $N = -U$ , and SOR is used as an outer iteration,

$$U_{SOR}^{(\nu)}(t) \;=\; U_{SOR}^{(\nu-1)}(t) \;+\; \omega\,(U_{Gauss-Seidel}(t) - U_{SOR}^{(\nu-1)}(t))\;. \qquad (5)$$

Unfortunately, these methods do not lead to a good acceleration. Reichelt has suggested in [4] to replace the *multiplication with the constant* $\omega$ by a *convolution with a function* $\omega(t)$,

$$U_{SOR}^{(\nu)}(t) \;=\; U_{SOR}^{(\nu-1)}(t) \;+\; \omega(t) \;\star\; (U_{Gauss-Seidel}(t) - U_{SOR}^{(\nu-1)}(t))\;. \qquad (6)$$

**Waveform GMRES.** ODE (1) is equivalent to an integral equation of the second kind:

$$(\mathcal{I} - \mathcal{K})\, U = \varphi\;. \qquad (7)$$

This non-self-adjoint operator equation can be solved by using Krylov subspace methods. In [2], Lumsdaine describes and implements a Waveform GMRES algorithm. He determines the $\nu$'th waveform iterate as the function $U^{(\nu)}$ that minimizes the norm of the residual, $||\; U^{(\nu)} - \mathcal{K}U^{(\nu)} - \varphi\;||$, in the Krylov subspace generated by the operator $\mathcal{I} - \mathcal{K}$ , i.e. the space spanned by $\{\varphi, \mathcal{K}\varphi, \mathcal{K}^2\varphi, \ldots, \mathcal{K}^{\nu-1}\varphi\}$. The computationally most expensive step in the algorithm is the evaluation of expressions of the form $(\mathcal{I} - \mathcal{K})\, U$. They can be evaluated by applying a standard waveform relaxation iteration to $U$. Note that $\mathcal{K}$, and, equivalently, the "preconditioning" of (7), depend on the splitting matrices $M$ and $N$.

**Multigrid waveform relaxation.** In the multigrid waveform relaxation method ([1, 5]), a coarse grid is used to accelerate the convergence of a standard waveform relaxation method. The algorithm is very similar to the multigrid algorithm for elliptic problems. Yet, all operators (smoothing, defect calculation, restriction, prolongation) operate on functions in time.

I will discuss the convergence of the algorithm for systems of ODEs derived by spatial discretization of the following model problems:

$$u_t = \epsilon\, u_{xx} \;+\; u_{yy} \quad \text{and} \quad u_t = \epsilon\,(u_{xx} \;+\; u_{yy}) + u_x + u_y\;. \qquad (8)$$

Standard point-wise red/black smoothing is no longer efficient for the small $\epsilon$ case. Robust waveform smoothers do exist, however, based on line-relaxation and/or semi-coarsening.

**4. Concluding remarks** The three methods described above have proven to be successful in accelerating the basic waveform relaxation method. This will be illustrated by examples from semi-discrete parabolic PDEs (heat equation, device simulation).

# References


[1] C. Lubich and A. Ostermann. Multigrid dynamic iteration for parabolic equations. *BIT*, 27:216–234, 1987.

[2] A. Lumsdaine. *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Values Problems, with Applications.* Ph.D.-thesis, M.I.T., dept of Electrical Engineering and Computer Science, January 1992.

[3] U. Miekkala and O. Nevanlinna. Convergence of dynamic iteration methods for initial value problems. *SIAM J. Sci. Stat. Comput.*, 8(4):459–482, July 1987.

[4] M. Reichelt. *Accelerated Waveform Relaxation Techniques for the Parallel Transient Simulation of Semiconductor Devices.* Ph.D.-thesis, M.I.T., dept of Electrical Engineering and Computer Science, June 1993.

[5] S. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems.* B.G. Teubner Verlag, Stuttgart, 1993.

# Space-time Iterative Methods for Time-Dependent Partial Differential Equations

Stefan Vandewalle
Caltech, Applied Mathematics 217-50, Pasadena, CA 91125

**1. Introduction.** We will address the problem of solving time-dependent partial differential equations on massively parallel processors. Classical time stepping methods are inherently sequential. This prohibits the use of massively parallel computers unless the problem on each time-level is very large. This observation has led to the development of algorithms that operate on more than one time-level simultaneously; that is to say, on grids extending in space and in time.

**2. Space-Time Iterative Methods.**

**Windowed Relaxation.** When a standard time-stepping method is implemented on a distributed memory multicomputer, each processor is assigned to a subset of the unknowns. At any given moment in the computation all processors are active on the same time-level. Local boundary information and intermediate results are exchanged by message passing.

In the windowed relaxation method by Saltz, Nicol and Naik ([4]), each processor executes a standard iterative method (Jacobi, Gauss-Seidel, SOR) *on a set of time-levels* before exchanging local boundary data. This leads to an efficient overlap of communication with calculation, and reduces message passing delays. Typical *windows* consist of two up to five time-levels. The use of larger windows is not recommended because of convergence degradation.

**Parallel Time-Stepping.** D. Womble in [7] assigns different processors to different time-levels. While processor 1 computes the solution on time-level $t_1$, processors 2, 3, ... $k$, update approximations to the solution on time-levels $t_2, t_3, \ldots t_k$. The method allows iterations on different time-levels to overlap, and works well when slowly convergent iterative methods are used. With multigrid however, in which case one iteration on every time-level is often sufficient, little or no overlap between time-levels is possible.

**Parabolic Multigrid.** We would like to treat the time-dimension as just another spatial dimension, and use multigrid to solve the set of equations on the entire space-time grid. The procedure requires a careful selection of the various multigrid operators: the smoothing method, the coarsening procedure and the restriction and prolongation operators. Three parabolic multigrid methods have appeared in the literature: the discrete-time multigrid waveform relaxation method ([5]), the time-parallel multigrid method ([1]), and the space-time multigrid method ([2]).

I will analyse the robustness of these methods with respect to the mesh aspect ratio $\Delta t/(\Delta x)^2$, where $\Delta t$ is the time-increment and $\Delta x$ is the spatial mesh width. It will be shown that the waveform method is very robust, and attains typical multigrid convergence rates independent of the spatial mesh size, the time-increment or the number of time-steps computed simultaneously. The time-parallel multigrid method is much less robust. Its use is restricted to grids where the time-increment is large compared to the fine grid spatial mesh size. The space-time multigrid method regains the robustness of the waveform method by using special restriction and interpolation operators, and a semi-coarsening strategy that depends on the value of the mesh aspect ratio.

**3. A Space-Time Direct Method.** Let the total number of grid points in space-time be $N$. It is a well-known information theoretic limit that the parallel complexity of any solver must by at least $O(\log(N))$. The space-time iterative solvers outlined above have a parallel complexity that is at best $O(\log^2(N))$ or $O(\log^3(N))$. I will show that an optimal method with $O(\log(N))$ complexity does indeed exist. The method is a direct solver based on the use of Fourier transforms in space, followed by a cyclic reduction type operation in time.

**4. Implementation and Timing Results.** These methods can be implemented on a message passing multicomputer by using a straightforward grid partitioning. Each process is assigned to a block of unknowns in the combined space-time grid. Extensive timing results illustrate a significant performance gain obtainable with the parabolic multigrid methods when compared to concurrent implementations of a variety of classic time-stepping solvers. In particular, speed-ups over 300 have been obtained on the Intel Delta ([6]). For a similar problem, solved on the same space-time grid to a similar accuracy, the speed-up of a standard time-stepping method was limited to about 20.

These methods are also very well suited for implementation on massively parallel systems of SIMD type. Assigning one processing element per grid point in the combined space-time domain leads to algorithms with extremely

low parallel complexities. The methods have been implemented on a 32K Connection Machine. Timing results illustrate and confirm the theoretically derived complexity estimates ([3]).

# References


[1] G. Horton. The time-parallel multigrid method. *Communic. in Appl. Num. Meth.*, 8:585–595, 1992.

[2] G. Horton and S. Vandewalle. A space-time multigrid method for parabolic P.D.E.s. Technical Report IMMD 3, 6/93, Universität Erlangen-Nürnberg, Martensstrasse 3, D-91058 Erlangen, Germany, July 1993. (to appear in SIAM J. Sci. Comput.).

[3] G. Horton, S. Vandewalle, and P. Worley. An algorithm with polylog parallel complexity for solving parabolic partial differential equations. Technical Report IMMD 3, 8/93, Universität Erlangen-Nürnberg, Martensstrasse 3, D-91058 Erlangen, Germany, July 1993. (to appear in SIAM J. Sci. Comput.).

[4] J. Saltz, V. Naik, and D. Nicol. Reduction of the effects of communication delays in scientific algorithms on message passing mimd architectures. *SIAM J. Sci. Stat. Comput.*, 8(1):s118–s138, January 1987.

[5] S. Vandewalle and R. Piessens. Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic partial differential equations. *SIAM J. Sci. Stat. Comput.*, 13(6):1330–1346, November 1992.

[6] S. Vandewalle and E. Van de Velde. Space-time concurrent multigrid waveform relaxation. *Annals of Numerical Mathematics*, 1(1-4):347–363, April 1994.

[7] D. Womble. A time-stepping algorithm for parallel computers. *SIAM J. Sci. Stat. Comput.*, 11(5):824–837, September 1990.

# Differential-Algebraic Equations in circuit analysis

M. van de Wiel
Philips Research

## Abstract

Historically, computer aided design of electrical networks stimulated researchers to study DAE's and their numerical solution. The BDF method for the solution of differential-algebraic systems as introduced by Gear in 1971 has become the standard method for time-domain simulation of electrical circuit models.

In general, an IC-designer will simulate the analogue behaviour of an electrical circuit in either the time domain or the frequency domain. We will restrict ourselves to the time-domain analysis as implemented in most circuit analysis tools, amongst others the Philips in-house circuit simulation tool Pstar.

In general, an electrical circuit design will consist of an irregular mixture of electrical connections and electrical components such as resistors, capacitors and transistors. Alternatively, the circuit design may be extended with a large number of parasitic components. The physical behaviour of each circuit component is accurately modelled by a set of one or more mathematical equations. The modelling of the complex nonlinear behaviour of transistors clearly illustrates the complexity of the overall circuit model. In the Modified Nodal Analysis modelling technique, a mathematical circuit model is constructed by enforcement of the Kirchhoff current conservation law at each common nodal point.

The equation systems originating from the application of the MNA method to a circuit design have some specific properties:

- sparse
- unstructured

- differential-algebraic
- (highly) nonlinear
- stiff

The Backward Differentiation Formulas (BDF) can be used to discretize the time derivatives in the differential-algebraic system. The resulting nonlinear systems are solved by a hybrid Newton-Raphson iteration: application of a.o. extrapolation, model bypassing, and relaxation results in an improved efficiency of the nonlinear solver. Within the Newton-Raphson iteration, linear systems are commonly solved by a direct method. Currently, research effort is being spent on the determination of an efficient iterative linear solver.

Since most DAE's encountered in circuit analysis are of index one, the BDF method proves to be efficient in daily use. Occasionally, DAE's of a higher index are encountered. In fact, DAE's of an arbitrary index can be constructed by cascading a series of circuits containing so called operational amplifiers. Consequently, additional measures have to be taken to guarantee accurate solution of higher index DAE's. In Pstar, this accuracy problem is solved by combining the time integration error control with an (optional) interpolation error control mechanism.

In the presentation an introduction will be given to the circuit analysis field. Special attention will be paid to the circuit modelling subject. In the second part of the presentation, the relation between DAE theory and practice will be discussed with the help of some example circuits. The main objective of the presentation will be to give the audience some insight in the background and complexity of circuit analysis.

# Automatic Data Structure Selection and Transformation for Sparse Matrix Computations (Extended abstract)

Harry A.G. Wijshoff
High Performance Computing Division
Department of Computer Science
Leiden University
P.O. Box 9512, 2300 RA Leiden
The Netberlands
harryw@cs.leidenuniv.nl

A significant part of scientific codes consists of sparse matrix computations which show notoriously bad efficiency on today's supercomputers. Mostly only a small fraction of the computing power of these computers can be utilized. There are many reasons for this. First, sparse matrix computations induce irregular data access which can significantly reduce memory bandwidth and cache utilization, e.g. spatial locality cannot always be exploited as efficiently. Secondly, the amount of possible reuse of data (temporal locality), is limited due to the fact that many operations are nullified making the ratio computations performed over the size of data sets very small. Not only prevents this data locality optimizations, also the communication overhead when these codes are executed on distributed memory computers can be substantial.
The third problem is caused by the fact that sparse matrices need to be represented in a compact way so that the *storage requirements and computational time* are kept to reasonable levels. This causes the representation of a sparse code in either FORTRAN with the occurrence of indirect addressing or in an other language with pointer structures to be very distorted. This is probably the most important problem of the ones listed above, because it does not only complicate software maintenance and the effort of producing sparse computation codes, but also most compiler optimizations get disabled.
The latter problem has been recognized for a long time now and there have been many efforts to overcome this problem. Most of the methods proposed in the past rely on pre-evaluation of the index sets of indirect addressed loops or provide runtime support to evaluate and reorder index sets. These methods can be successfully employed, but only in the cases were index sets are constant during a period of time, like for instance in triangular solvers. However, as index sets change during the course of loop execution these methods will fail due to the incurred overhead of this pre-evaluation

or runtime evaluation. Another problem of these methods is that the data structure as selected by the programmer cannot be changed. As different architectural features might favour different data structures for sparse matrices, the code will stay inefficient regardless of the optimizations performed if the sparse code is not specifically targeted for the architectural features of a particular machine. So, in effect, to overcome the problem of sparse matrix computations the compiler should not only be able to perform program transformations but also transformations on data structures themselves.
In order to deal with the data structure problem of sparse computations let us first step back for a moment and define what we actually mean by sparse computations. The simplest and the most generic definition is: *a sparse computation is a computation which computes on sparse data structures and a sparse data structure (i.e. a sparse matrix) is a data structure which is logically contained in an enveloping data structure (i.e. a dense matrix).* The underlying problem for sparse computations now is *where* to deal with the fact that only part of the enveloping data structure is computed on. The common approach is to deal with this at the algorithm (programming) level. However, why should it not be possible to deal with this issue at a lower level, i.e. at the compilation level.
This implies that the computation is defined on the enveloping data structure, i.e. dense matrix, and the compiler transforms the code to execute on the sparse data structure. This has the advantage that the compiler does not need to extract program knowledge from an obscured code, but is presented with a much cleaner program on which regular data dependence checking and standard optimizations can be performed. The other advantage is that the compiler performs the final data structure selection, which can be done after standard optimizations are performed, or at the same time when deciding which program transformations to perform. Due to the phase ordering problem the method as presented in this paper will rely on data structure selection after program transformations are performed. Feedback will be given to the transformation phase when data structure selection cannot be resolved correctly (efficiently). The initial program defined on the enveloping data structure will be saved. This enables the compiler after having performed the code and data structure transformations for a particular architecture to retarget the code to an other machine starting from the original code.
A user defined annotation is used to identify which of the declared dense data structures of the program are actually sparse. This annotation only needs to occur in the declarative part of the program and as such is not intrusive. The compiler can also prompt the programmer with the data structures declared in the program and inquire explicitly about the sparsity of each of them. After the sparse data structures are identified the compiler might need specific information about the sparsity structure of the data structures, e.g. the density or bandwidth, to make efficient data structure selection. This can be obtained by interactive inquiries, user defined annotation, or by automatic evaluation of these characteristics using a library function which reads the sparse data structure from file, and performs the

necessary statistics. Note that if the sparse data structure is on file then specific input/output routines have to be inserted in the code.
The data structure selection and transformation method as described in this talk is based on a *data driven* approach and consists of a three phase process. In the first phase the instructions in the code are identified which will be affected by the sparsity of the data structure and the code will be optimized to make fully use of these statements. In the second phase the data structure selection is performed and conflicts are resolved. In the third phase the actual transformations are performed to generate the resulting code.

**List of Participants**

| | | | | |
|---|---|---|---|---|
| 1. | A.C. | Berkenbosch | TUE | Eindhoven |
| 2. | H. | Bijl | TUD | Delft |
| 3. | J.G.L. | Booten | CWI | Amsterdam |
| 4. | M.J.A. | Borsboom | WL | Marknesse |
| 5. | E. | Brakkee | TUD | Delft |
| 6. | K. | Dekker | TUD | Delft |
| 7. | Th.J. | Dekker | | Castricum |
| 8. | A.C.N. | van Duin | RUL | Leiden |
| 9. | P.J.J. | Ferket | TUE | Eindhoven |
| 10. | D.R. | Fokkema | RUU | Utrecht |
| 11. | J.F. | Frankena | UT | Enschede |
| 12. | C. | Führer | DLR | Wessling |
| 13. | M.B. | van Gijzen | RUU | Utrecht |
| 14. | J.A. | van de Griend | RUL | Leiden |
| 15. | P.P.N. | de Groen | V.U. Brussel | Brussel |
| 16. | J. | de Groot | Philips Res. Lab. | Eindhoven |
| 17. | E. | Hairer | Univ. de Genève | Genève |
| 18. | W. | Hoffmann | UvA | Amsterdam |
| 19. | K.J. | in 't Hout | Stieltjes/CWI | Leiden/Amsterdam |
| 20. | P.J. | van der Houwen | CWI | Amsterdam |
| 21. | W.H. | Hundsdorfer | CWI | Amsterdam |
| 22. | J.K.M. | Jansen | TUE | Eindhoven |
| 23. | W. | Joppich | GMD | Sankt Augustin |
| 24. | E.F. | Kaasschieter | TUE | Eindhoven |
| 25. | D.E. | Keyes | ICASE | Hampton |
| 26. | J. | Kok | CWI | Amsterdam |
| 27. | M. | Kooper | TUD | Delft |
| 28. | J.F.B.M. | Kraaijevanger | KSEPL | Rijswijk |
| 29. | M.E. | Kramer | KSLA | Amsterdam |
| 30. | W.M. | Lioen | CWI | Amsterdam |
| 31. | R.M.M. | Mattheij | TUE | Eindhoven |
| 32. | M.G. | Neytcheva | KUN | Nijmegen |
| 33. | M. | Nool | CWI | Amsterdam |
| 34. | M.J. | Noot | TUE | Eindhoven |
| 35. | M.H.C. | Paardekooper | KUB | Tilburg |
| 36. | H.J.J. | te Riele | CWI | Amsterdam |
| 37. | P. | Smit | KUB | Tilburg |
| 38. | B.P. | Sommeijer | CWI | Amsterdam |
| 39. | E.J. | Spee | CWI | Amsterdam |
| 40. | S.P. | Spekreijse | NLR | Marknesse |
| 41. | M.N. | Spijker | RUL | Leiden |
| 42. | Th.L. | van Stijn | RIKZ | Den Haag |
| 43. | W.J.H. | Stortelder | CWI | Amsterdam |
| 44. | F.A.J. | Straetemans | RUL | Leiden |
| 45. | J.J.B. | de Swart | CWI | Amsterdam |
| 46. | K.H. | Tan | RUU | Utrecht |
| 47. | C. | Traas | UT | Enschede |
| 48. | S. | Vandewalle | Caltech | Pasadena |
| 49. | W.A. | van der Veen | CWI | Amsterdam |
| 50. | M. | van Veldhuizen | VU | Amsterdam |
| 51. | A.E.P. | Veldman | RUG | Groningen |
| 52. | J.G. | Verwer | CWI | Amsterdam |

| | | | | |
|---|---|---|---|---|
| 53. | G.A.L. | van de Vorst | TUE | Eindhoven |
| 54. | P.W.C. | Vosbeek | TUE | Eindhoven |
| 55. | C. | Vuik | TUD | Delft |
| 56. | P. | Wesseling | TUD | Delft |
| 57. | M.J.C. | van de Wiel | Philips Res. Lab. | Eindhoven |
| 58. | P.M.E.J. | Wijckmans | TUE | Eindhoven |
| 59. | H.A.G. | Wijshoff | RUL | Leiden |
| 60. | P. | Wilders | TUD | Delft |
| 61. | P.M. | de Zeeuw | CWI | Amsterdam |
| 62. | P.A. | Zegeling | RUU | Utrecht |
| 63. | M. | Zijlema | TUD | Delft |