52

Parallel Solvers for the Two-Group Neutron Diffusion Equations of Reactor Kinetics

ROBERT SCHEICHL[1]

INTRODUCTION

We are concerned with the solution of the transient two-group neutron diffusion equations arising in the simulation process of nuclear reactor cores [DH76]:

$$\frac{\partial \phi_g}{\partial t} = \vec{\nabla} \cdot (D_g \vec{\nabla} \phi_g) + f_g \left(\phi_1, \phi_2, (C_i)_{i=1,\ldots,6} \right), \qquad g = 1, 2 \tag{1}$$

$$\frac{\partial C_i}{\partial t} = h_i(\phi_1, \phi_2, C_i), \qquad i = 1, \ldots, 6 \tag{2}$$

The unknowns in this system are the flux densities ϕ_1 and ϕ_2 of the "fast" ($g = 1$) and "thermic" ($g = 2$) neutrons, as well as the precursor concentrations C_i in each precursor group $i = 1, \ldots, 6$. The linear operators f_g and h_i, and the diffusion coefficients D_g are determined by the underlying physics. We consider this system on a polyhedral domain $\Omega \subset R^3$, with given initial data at $t = t_0$ and the following vacuum boundary condition on $\partial\Omega$:

$$\left(\frac{\partial \phi_g}{\partial \vec{n}} \right)_{in} = \vec{0}, \qquad g = 1, 2. \tag{3}$$

In this paper we investigate the excellent parallel performance of preconditioned Bi-CGStab and Multi-Grid applied to this system of equations. Other than reported

1 Dept. Math. Sci., University of Bath, UK. Email: R.Scheichl@maths.bath.ac.uk.
This research was supported by the Siemens AG, and a Leonardo da Vinci scholarship.
Eleventh International Conference on Domain Decomposition Methods
Editors Choi-Hong Lai, Petter E. Bjørstad, Mark Cross and Olof B. Widlund

elsewhere in the literature (e.g. [FB+88, HP+96, HDB96, UTK95, YN93]), we are able to achieve optimal (linear) speedup for all tested examples up to 12 processors on all but the coarsest grid. Moreover, the parallel execution times are better than the parallel execution times of the established reactor simulation code PANBOX of the Siemens AG Operating Division KWU, described in [FB+88] and [HP+96]. This is of practical interest, especially since the multilevel method implemented in PANBOX is one of the most efficient methods for the sequential case to date.

We use a special finite volume method (NEM-M0), and a method that combines Crank-Nicholson and the BDF(2)-method to discretise Equations (1-3) in space and time, respectively. The parallelisation is based on a simple grid partitioning. The most important features of our methods are firstly the spatial discretisation that assigns each degree of freedom to exactly one subdomain and therefore needs no assembling at the subdomain interfaces, and secondly an effective parallel hybrid Block-Jacobi - Block-SOR smoother / preconditioner that makes use of the particular structure of the linear equation systems. The implementation of the methods was carried out on a shared memory 12-processor machine (SGI Power Challenge) using "threads". For more details about the methods and the implementation, and for more numerical results see [Sch97].

DISCRETISATION

On a uniform rectangular grid $\mathcal{T} = \{T_m : m = 1...M\}$ we discretise Equations (1-3) by using the mixed cell-centred finite volume method NEM-M0, especially proposed in [FBW77] for the multigroup neutron diffusion equations. It is closely related to lowest-order Raviart–Thomas–Nédélec mixed-hybrid finite elements [HDV93]. To simplify the presentation let $T_0 = R^3 \backslash \Omega$. As for mixed finite elements, we introduce the physically important neutron current density

$$\vec{j}_g = -D_g \vec{\nabla} \phi_g, \qquad g = 1, 2 \qquad \text{(Fick's Law)}. \tag{4}$$

To obtain semi-discrete (time dependent) approximations of Equations (1-2) we introduce zeroth momenta ϕ_g^m and C_i^m of the unknown functions ϕ_g and C_i on each element T_m, like in standard finite volume approximations. The vector-valued Equation (4) on the other hand, is integrated over each face $F_{m_1,m_2} = \bar{T}_{m_1} \cap \bar{T}_{m_2} \neq \emptyset$ of the triangulation $\mathcal{T}$, leaving us with an equation for the average current density $j_g^{m_1,m_2}$ on face F_{m_1,m_2}. The significant step is the separation of this average current density into a current $\vec{j}_g^{\,m_1+}$ from the left element T_{m_1} and a current $\vec{j}_g^{\,m_2-}$ from the right element T_{m_2}. Let $u \in \{x, y, z\}$ denote one of the coordinate directions, and let F_{m_1,m_2} be perpendicular to u. Then

$$j_g^{m_1,m_2} = \left(j_g^{m_1+}\right)_u - \left(j_g^{m_2-}\right)_u . \tag{5}$$

The boundary condition (3) can then be easily fulfilled by setting $\vec{j}_g^{\,0\pm} = \vec{0}$. After a simple linear transformation of the resulting system we get the following differential-algebraic equations:

$$\frac{d\phi_g^m}{dt} = f_g^m\left(\phi_1^m, \phi_2^m, \left(j_g^{l_u+}\right)_{u\in\{x,y,z\}}, \left(j_g^{r_u-}\right)_{u\in\{x,y,z\}}, \left(C_i^m\right)_{i=1,\dots,6}\right) \quad (6)$$

$$\frac{dC_i^m}{dt} = h_i^m(\phi_1^m, \phi_2^m, C_i^m) \quad (7)$$

$$\left(j_g^{m\pm}\right)_u = k_{gu}^{m\pm}\left(\phi_g^m, \left(j_g^{l_u+}\right)_u, \left(j_g^{r_u-}\right)_u\right) \quad (8)$$

The operators f_g^m, h_i^m, and $k_{gu}^{m\pm}$ are all linear again, and l_u (resp. r_u) denotes the left (resp. right) neighbour of m in u-direction.

The system of differential Equations (6-7) is stiff and it is necessary to use stable implicit methods for its integration. As a first-order method we use implicit backward–Euler, as a second-order method we use TR-BDF(2), a composition of Crank-Nicholson with the second-order backward differentiation formula proposed in [BC+85]. This method combines the desirable properties of both methods. It is as stable as BDF(2), but it has a smaller truncation error.

The integration of (7) leads to an explicit equation for C_i^m, and C_i^m can be substituted in (6). Integrating (6) for all energy groups g and elements m and coupling it with (8) finally leads to a $(14\,M)$–dimensional non-symmetric linear equation system

$$A\,\mathbf{x} = \mathbf{b} \quad (9)$$

for $\mathbf{x} := \left(\phi_{\mathbf{1}}^{\mathbf{1}}, \phi_{\mathbf{2}}^{\mathbf{1}}, (\mathbf{j_1^{1-}})_{\mathbf{x}}, \dots, (\mathbf{j_2^{1+}})_{\mathbf{z}}, \dots\dots, \phi_{\mathbf{1}}^{\mathbf{M}}, \phi_{\mathbf{2}}^{\mathbf{M}}, (\mathbf{j_1^{M-}})_{\mathbf{x}}, \dots, (\mathbf{j_2^{M+}})_{\mathbf{z}}\right)$ in each time step.

PARALLEL SOLVERS

We solve these non-symmetric linear equation Systems (9) using standard Multi-Grid [Hac85] and left-preconditioned Bi-CGStab (P-BiCGStab) [VdV92] on grids of different fineness, where the fine grids are obtained by uniform refinement of the coarse-grid in each coordinate direction. The key components of Multi-Grid are as follows: the coarse-grid correction is obtained using a W-cycle ($\gamma = 2$); the systems on the coarsest grids are solved iteratively using P-BiCGStab; the prolongation P is chosen to be piecewise constant for the fluxes ϕ_g^m, and piecewise linear for the currents $\vec{j}_g^{m\pm}$; the restriction $R = 1/8\ P^T$ using the Galerkin approach; the smoother is a hybrid Block-Jacobi - Block-SOR method that will be discussed in more detail at the end of this section. If we use μ (resp. ν) pre- (resp. post-) smoothing steps in the Multi-Grid method, we will denote the method by MG(μ, ν). We will also use this hybrid method for the preconditioning of Bi-CGStab.

All these solution methods can be fully described in terms of vector operations and matrix-vector multiplications. To parallelise these operations we have to partition the matrices and vectors. We partition the grid(s) $\mathcal{T} = \{T_m : m = 1, \dots, M\}$ into evenly sized parts $\mathcal{T}_i$ (homogeneous parallelisation), and assign each part and all unknowns and equations that are defined thereon to one processor. The advantage that the NEM-M0 discretisation has over other discretisations is that each component of a vector can be assigned to exactly one element T_m and therefore to one part $\mathcal{T}_i$ of the grid. An important consequence of this property is that vectors do not have to be assembled on the interfaces between two parts of the grid. In the following we will talk of local

calculations, if we are concerned only with calculations on one part $\mathcal{T}_i$ of the grid. The local parts of a matrix A and a vector $\mathbf{x}$ on $\mathcal{T}_i$ will be denoted by A_i and $\mathbf{x_i}$ respectively.

For the basic vector operations the individual computations for each component are independent and can be carried out entirely in parallel. In scalar products and matrix-vector multiplications on the other hand, data dependencies arise, and communication or synchronisation is needed. Scalar products $\sigma = \langle \mathbf{x}, \mathbf{y} \rangle$ can be parallelised by reduction. In a first step local scalar products $\sigma_i = \langle \mathbf{x_i}, \mathbf{y_i} \rangle$ are calculated on each processor. In a "critical section" in which the calling processor has sole access to the statements contained within it, they are added to the global scalar product σ. Any other processor, on encountering the occupied section, is forced to wait until the section is cleared. For the matrix-vector multiplications on the other hand, in particular for $A \cdot \mathbf{x}$ and the restriction $R \cdot \mathbf{x}$, we need synchronisation. Each processor needs components of the vector $\mathbf{x}$ that are assigned to its neighbouring processors. Since the processors have shared memory access, we only have to synchronise them at the beginning of each matrix-vector multiplication to guarantee that.

Let us finally discuss the smoother in MG(μ, ν) and the preconditioner for Bi-CGStab. The matrix A in (9) is built up by 14×14 diagonal blocks A_{mm} and a small number of off-diagonal entries. The diagonal blocks have the following structure

$$A_{mm} = \left(\begin{array}{cc|c} * & * & \\ * & * & \mathbf{0} \\ \hline * & 0 & \\ \vdots & & I_{12} \\ 0 & * & \end{array} \right), \tag{10}$$

and can be inverted fast and easily. In the sequential case this fact can be exploited in an effective Block-SOR smoother/preconditioner. However, the solution of each block system in the Block-SOR method always depends on the solution of the previous blocks. A straightforward parallelisation of the Block-SOR method is therefore not effective, since the processors would have to wait for each other. The natural way to avoid this, is the application of a hybrid Block-Jacobi - Block-SOR method. Locally each processor uses one iteration of the original Block-SOR method for the approximate inversion of the diagonal block of his part A_i of A, and calculates the local components of the next iterate using Block-Jacobi relaxation. These operations to find the next iterate are independent of the calculations on all other processors, and can be done entirely in parallel. It is only necessary to synchronise the processors prior to each iteration. If the number of elements T_m per part $\mathcal{T}_i$ is reasonably high, the hybrid method only differs in very few components from the sequential method.

NUMERICAL RESULTS

We applied both solution methods in the simulation process of real three-dimensional reactor problems to test their robustness, speedup, and efficiency on three grids of different fineness. The numbers of unknowns N of the systems that arise in each time step are about $5.5 \cdot 10^4$, $4.5 \cdot 10^5$, and $3.5 \cdot 10^6$ on the three different grids. Finer grids were not possible because of a lack of computer memory. Coarser grids are hard to

introduce because of the difficult geometry of a reactor. It was therefore only possible to run MG(μ, ν) on the two finer grids. If not otherwise stated, the results have all been obtained using TR-BDF(2) as the time discretisation. One composite time step therefore involves the solution of two linear equation systems.

Robustness and comparison of MG(3/2) and P-Bi-CGStab

The methods converge robustly and the parallelisation has no effect on the convergence, for all the tested problems. In Figures 1 and 2 the iterations and execution times for a short transient of one of the examples on the second grid are given. The time steps are determined adaptively. To check for convergence we use the residual test with $\varepsilon = 10^{-6}$. The iterations and the execution times are almost constant after a short start-up period. In comparison, MG(3/2) is about 2.5 times faster than P-Bi-CGStab throughout the simulation for this problem. It is important to note that the modification of the Block-SOR method in the parallel case does not effect the convergence of the algorithms significantly, and the iterations are in average not higher than in the sequential case.

More generally (see [Sch97]), we can conclude that for the tested examples MG(3/2) is about 2 to 3 times faster than P-BiCGStab on the second grid, and about 3 to 4 times faster on the finest grid. The convergence of both methods does not depend on the partitioning. For MG(3/2) the rate of convergence lies between 0.3 and 0.5 for all examples, and does not depend strongly on the number of unknowns N. However, for a more thorough investigation of the asymptotics we would need more results for other values of N. For P-Bi-CGStab, on the other hand, the rate of convergence increases from 0.5 on the coarsest grid to 0.85 on the finest. Since one iteration of MG(3/2) is only 20 % more expensive than P-Bi-CGStab on all grids we can conclude as expected: the finer the grid becomes the more MG(3/2) outperforms P-Bi-CGStab.

Speedup

The speedup of both methods is optimal (linear) for all tested examples on all but the coarsest grid. On the coarsest grid the speedup for P-Bi-CGStab is also optimal, if less than 10 processors are used. For a higher number of processors the speedup curve starts to level off slightly. In Figure 3 the speedup graphs for one of the examples are shown. We can see the claimed excellent behaviour. The reasons for the superlinear speedup on the second grid are different numbers of iterations and cache-effects. Moreover, these results suggest a good performance even for a much higher number of processors which were unfortunately not available.

Comparison to other parallel reactor simulation codes

An exact comparison of the speedup results with the results reported in the literature [FB+88, HDB96, UTK95, YN93] is difficult, since they are all obtained on different computer systems, for different examples, and for different problem sizes. Nevertheless, none of them obtains optimal speedup for full three-dimensional problems. The best comparison are the results in [HP+96]. They were obtained on exactly the same computer for the same examples, also using threads for the parallelisation of the

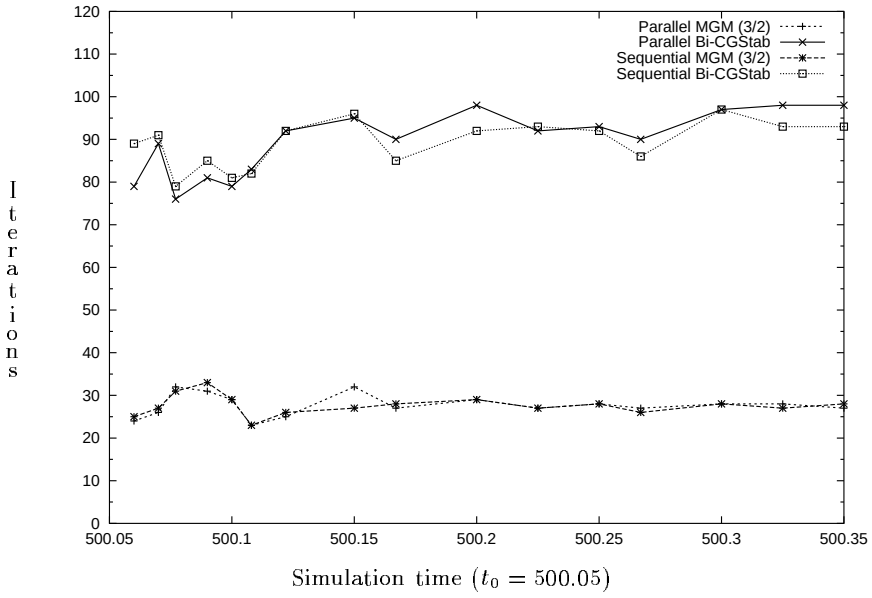

Figure 1 Number of iterations (parallel: 9 processors)

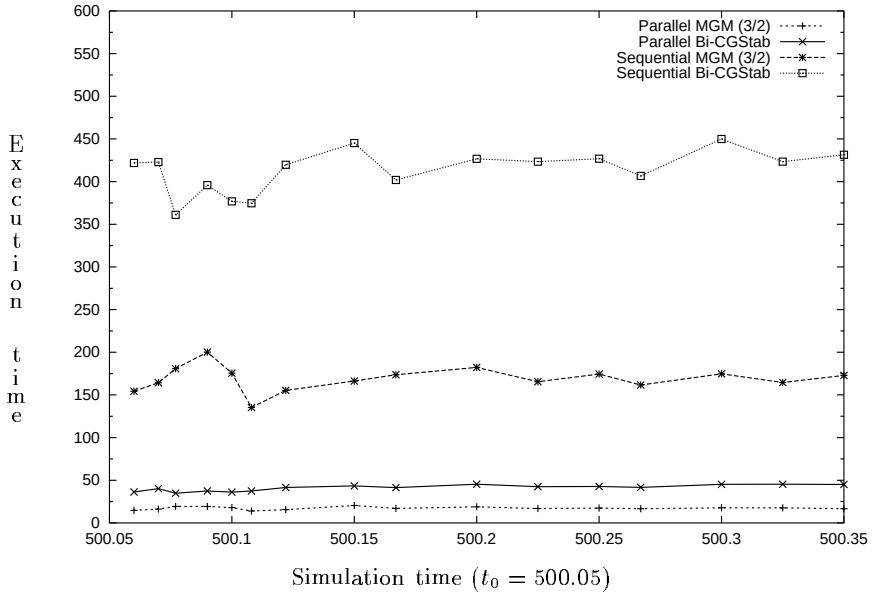

Figure 2 Execution time (in seconds) (parallel: 9 processors)

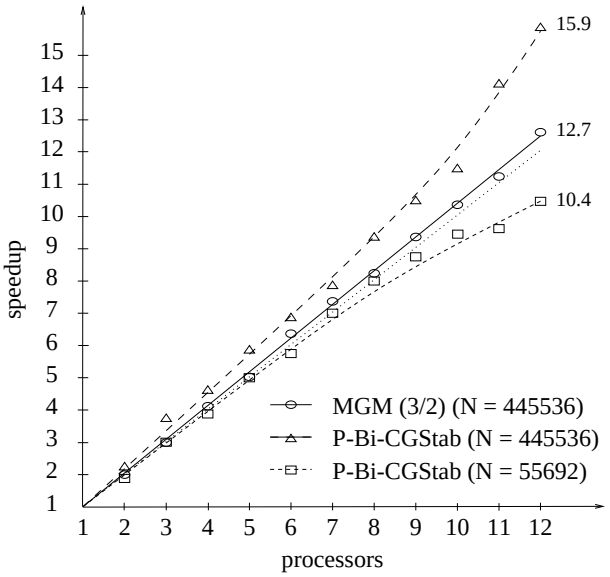

Figure 3 Speedup for one time step ($dt = 0.01$ s)

established reactor simulation code PANBOX of the Siemens AG Operating Division KWU. On the coarsest grid they report a speedup of 2.7, 3.5, and 3.7 for 4, 6, and 8 processors respectively (see Fig. 7 in [HP+96]). Even for higher numbers of processors the speedup will not be much higher than 4. The corresponding speedup factors of P-Bi-CGStab are 3.8, 5.7 and 8.0, and the speedup can be increased to more than 10 for higher numbers of processors. On a finer grid ($N = 2.2 \cdot 10^5$) the speedup factors of PANBOX are 4.1, 5.8, and 7.3 for 4, 6, and 8 processors respectively. In comparison to the speedup obtained with MG(3/2) and P-Bi-CGStab on the second grid, the speedup curve of PANBOX seems to start to level off already at 8 processors.

For the comparison of the execution times we calculated 10 time steps of a control rod ejection benchmark problem on the coarsest grid using PANBOX and P-Bi-CGStab. This is of practical interest, since PANBOX is one of the most efficient methods for the sequential case to date. The solution of 10 time steps of the benchmark problem takes 53 seconds using PANBOX on the SGI Power Challenge. In PANBOX, Equations (1-3) are discretised using NEM-M0 and a special first-order heuristic exponential integration scheme related to implicit backward-Euler. Using P-Bi-CGStab to calculate the same 10 time steps of the benchmark problem with the implicit backward-Euler method takes 115 seconds on the same machine. For a fair comparison we also used the same convergence test as in PANBOX. We see that the sequential version of P-Bi-CGStab is about two times slower than PANBOX. Nevertheless, the far better speedup of P-Bi-CGStab makes it possible to catch up with PANBOX already for 8 processors. For higher numbers of processors, P-Bi-CGStab is therefore going to be faster than the established reactor simulation code PANBOX.

ACKNOWLEDGEMENTS

The author would like to thank Prof. U. Langer from the University of Linz, Dr. M. Paffrath from Siemens ZT Munich, and Dr. I. Graham from the University of Bath for their helpful advice and comments.

REFERENCES

[BC+85] Bank R., Coughran W. M., *et al.* (1985) Transient simulation of silicon devices and circuits. *IEEE Transactions on Computer Aided Design of integrated circuits* 4(4): 436–451.

[DH76] Duderstadt J. J. and Hamilton L. J. (1976) *Nuclear reactor analysis.* J. Wiley & Sons, New York.

[FB+88] Finnemann H., Brehm J., *et al.* (1988) Mulitgrid solution of diffusion equations on distributed memory multiprocessor systems. *Atomkernenergie / Kerntechnik* 52(3): 169–174.

[FBW77] Finnemann H., Bennewitz F., and Wagner M. R. (1977) Interface current techniques for multidimensional reactor calculations. *Atomkernenergie / Kerntechnik* 30: 123–128.

[Hac85] Hackbusch W. (1985) *Multi-grid methods and applications.* Springer, Berlin.

[HDB96] Han G. Joo, Downar T. J., and Barber D. A. (1996) Methods and performance of a parallel reactor kinetics code PARCS. *Proceedings Int. Conf. Physics of Reactors* pages J42–J51.

[HDV93] Hennart J. P. and Del Valle E. (1993) On the relationship between nodal schemes and mixed-hybrid finite elements. *Numerical Methods for Partial Differential Equations* 9: 411–430.

[HP+96] Hammer C., Paffrath M., *et al.* (1996) Performance of the coupled thermalhydraulics/neutron kinetics code R/P/C on workstation clusters and multiprocessor systems. *Proceedings Int. Conf. Physics of Reactors* .

[Sch97] Scheichl R. (1997) *Parallel solution of the transient multigroup neutron diffusion equations with multi-grid and preconditioned Krylov-subspace methods.* (Master's Thesis), volume C 21 of *Schriften JKUL.* Trauner, Linz.

[UTK95] Uematsu M., Tsuiki M., and Kubota K. (1995) Parallel processing of boiling reactor core analysis. *Journal Nuclear Science and Technology* 32(6).

[VdV92] Van der Vorst H. A. (1992) Bi-CGSTAB: A more smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 13: 631–644.

[YN93] Yong H. Kim and Nam Z. Cho (1993) Parallel solution of the neutron diffusion equation with the domain decomposition method on a transputer network. *Nuclear Science and Engineering* 114: 252–270.