Performance of preconditioners for the distributed vector finite-element time-domain algorithm

Alain Nicolas, Laurent Nicolas, Christian Vollaire, Boguslaw Butrylo

Performance of Preconditioners for the Distributed Vector Finite-Element Time-Domain Algorithm

Alain Nicolas[1], Laurent Nicolas[1], Christian Vollaire[1], and Boguslaw Butrylo[2]

[1]Centre de Genie Electrique de Lyon (CEGELY) UMR CNRS 5005, Ecole Centrale de Lyon, 69134 Ecully, France
[2]Bialystok Technical University, Faculty of Electrical Engineering, 15-351 Bialystok, Poland

This paper deals with some aspects of performance of the symmetric successive over-relaxation preconditioner in a distributed environment. The details of distributed formulation of the preconditioner are presented. Some performance metrics are compared and discussed for the message passing interface implementation of the algorithm. The properties of the solver are estimated for concurrent three-dimensional formulation of the finite-element time-domain method. The analyzed benchmark models are approximated by tetrahedral first order Whitney elements.

***Index Terms*—Edge elements, finite-element (FE) method, iterative solver, parallel numerical algorithms, time-domain algorithm.**

I. Introduction

WIDE acceptance of distributed processing in modeling and optimization of electromagnetic phenomena requires an efficient solver. The overall numerical performance of the solver, computational cost, and flexibility of the distributed implementation of the algorithm are the basic issues in efficient computations.

The general advantages of the conjugate gradient (CG) algorithm make it very useful in computational electromagnetics. The efficiency of the method is improved by implementation of different types of preconditioners [1]–[3]. The final formulation and detailed properties of the distributed solver are determined by at least three factors:

- — the mathematical formulation of the method (i.e., structure of subtasks and number of independent threads);
- — the method of parallelization (i.e., the method of data and/or task decomposition);
- — the technical profile of multicomputer hardware platform.

The objective of this paper is to present the distributed formulation of the symmetric successive over-relaxation (SSOR) preconditioner implemented in the preconditioned conjugate gradient (PCG) algorithm [4]–[6]. The presented algorithm is based on the single processing multiple data (SPMD) paradigm. To evaluate the performance gain, the solver is executed on a distributed processing system, which consists of a collection of interconnected stand-alone computers.

II. Distributed PCG Algorithm With SSOR Preconditioner

The large-scale time-domain analysis is a computationally demanding task [7]. The most computationally expensive part of the algorithm and the hardest problem in the distributed implementation is the solver of matrix equation derived from finite-element (FE) formulation [8], [9]. The spatial discretization of an analyzed electromagnetic problem yields a matrix equation $\mathbf{A} \cdot \mathbf{e}_n = \mathbf{b}_n$. The presented algorithm concerns time-domain formulation, therefore $\mathbf{e}_n$ represents the time-dependent distribution of electric field, and the $\mathbf{A}$ matrix is real, positive definite, and well conditioned, $\dim(\mathbf{A}) = N_{\mathrm{DOF}} \times N_{\mathrm{DOF}}$.

The presented formulation of the SSOR preconditioner is based on the task decomposition and overlapped distributed processing. The task structure and interdependencies between concurrent threads (including forward calculation, calculation of a diagonal matrix, and backward substitution) arise from the implemented method of matrix decomposition and the data storage scheme. The computational cost of the solver is determined by these two factors.

The presented form of the algorithm is based on the compressed row storage (CRS) form of the $\mathbf{A}$ matrix [5]. Therefore, the row-wise matrix decomposition is applied. The number of submatrices $\mathbf{A}_1, \ldots, \mathbf{A}_n, \ldots, \mathbf{A}_N$ is equal to the total number of processing workstations N, and $\dim(\mathbf{A}_n) = (N_{\mathrm{DOF}}/N) \times N_{\mathrm{DOF}} = N_{\mathrm{DOF},n} \times N_{\mathrm{DOF}}$.

The parallelism of the SSOR preconditioner is achieved by changing the structure of subtasks in the algorithm. The mathematical formulation of the forward and backward steps forces different decomposition of tasks. The matrix is divided by columns in the forward calculation, while in the backward substitution the subtasks are defined by rows. The $\mathbf{u}$ and $\mathbf{v}$ vectors are the final results of the forward calculation step and the backward substitution step, respectively. Both vectors are split in the distributed version $\mathbf{u} = \mathbf{u}_1 \cup \cdots \cup \mathbf{u}_N$, and $\mathbf{v} = \mathbf{v}_1 \cup \cdots \cup \mathbf{v}_N$, and $\dim(\mathbf{u}_n) = \dim(\mathbf{v}_n) = N_{\mathrm{DOF},n}$.

The second level of concurrency of the algorithm consists in the decomposition of the local submatrices $\mathbf{A}_n$. In this way, the granularity of the algorithm (i.e., the relation between concurrent and sequential parts) can be matched. The granularity of the algorithm is described by the integer number $G(= 1, 2, 3, \ldots)$. In this case, the whole structure the algorithm and elementary tasks concern some elementary matrices $\mathbf{A}_{n,i}, \dim(\mathbf{A}_{n,i}) = (N_{\mathrm{DOF},n}/G) \times N_{\mathrm{DOF}}$. Large value of the G number corresponds with a fine-grained problem formulation. The computing units perform relatively small number of floating-point operations between data transfers. Unfortunately, in this way, the communication load increases.

Digital Object Identifier 10.1109/TMAG.2005.846055

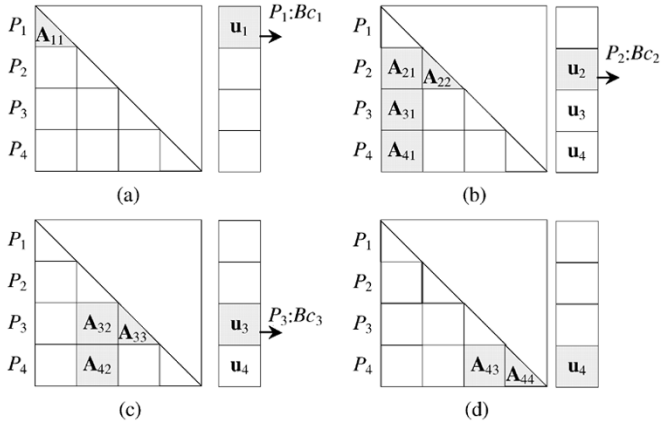

Fig. 1. Sequence of matrix operations in the forward stage of the preconditioner ($N = 4, G = 1$). The $\mathbf{A}_{ij}$ submatrices and $\mathbf{u}_i$ subvectors denote active parts of the data sets. Arrows indicate the broadcasting workstation and Bc_i is the symbol of the collective communication command.

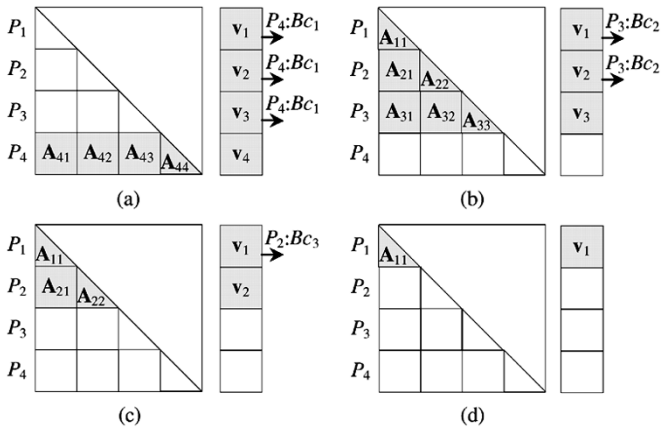

Fig. 2. Sequence of matrix-vector calculations in the backward stage ($N = 4, G = 1$).

The forward calculation step (including matrix-vector product) is the hardest part of the SSOR algorithm in this formulation (Fig. 1). The total number of data transfers N_B depends on the number of processing workstations and the granularity of domain decomposition $N_B = (N - 1) \cdot G$. The backward step is not constrained by the implemented CRS scheme and the matrix decomposition. The number of data transfers is directly proportional to the number of workstations $N - 1$ (Fig. 2).

The method of domain decomposition is reflected in the structure of the algorithm and in the mutual dependencies of the processes. Fig. 3 depicts the time line of a single course of the presented concurrent SSOR preconditioner. The sequence of independent data processing and communication tasks for $P_1, P_2, \ldots$, and P_N computing units are ordered along vertical lines. The rhombus represent data transfers commands (either input or output), while the rectangles describe independent data processing tasks. The direction of a horizontal arrow determines the broadcasted processing unit.

The inherently sequential nature of some parts of the SSOR algorithm limits parallelism of the distributed implementation. Only the intermediate level (calculation of the diagonal matrix) works in fully parallel mode (no data transfers between processing units). The tasks in the forward and the backward steps are partially overlapped.

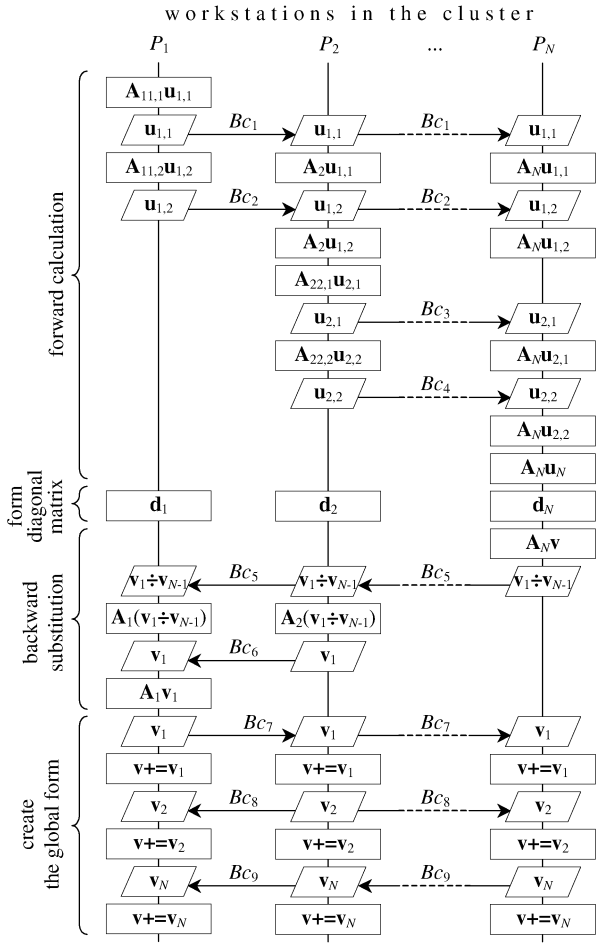

Fig. 3. Time line of the distributed SSOR preconditioner ($G = 2$). Arrows indicate the broadcasting workstation and Bc_i is the symbol of the collective communication command.

III. Test Problem and Distributed Environment

The presented form of the PCG solver with SSOR preconditioner is used to calculate different size test problem. The properties of the algorithm are discussed for a time-domain analysis of an open boundary high frequency electromagnetic benchmark problem. The presented algorithm is used to calculate a problem with a monochromatic plane wave propagating in free space. The final matrix equation is derived from the Maxwell's equations for some linear and isotropic media.

The forms of the $\mathbf{A}$ matrix and $\mathbf{b}_n$ vector arise from the implemented time integration scheme [10]. Since the conditionally stable central Euler time integration schemes is used, the coefficients of the linear matrix $\mathbf{A}$ are stated by equation

$$a_{ij} = \int_V \left(\varepsilon + \frac{\Delta t \sigma}{2}\right) \mathbf{W}_i \mathbf{W}_j dV + \frac{\Delta t}{2} \int\limits_{S_{\mathrm{ABC},p}} \mathbf{W}_j \frac{1}{\mu c} (\mathbf{W}_i \times \vec{\mathbf{n}})\, dS \quad (1)$$

where $i, j = 1, \ldots, N_{\mathrm{DOF}}$, and $\mathbf{W}$ is the vector shape function because the model is approximated by the first order incomplete

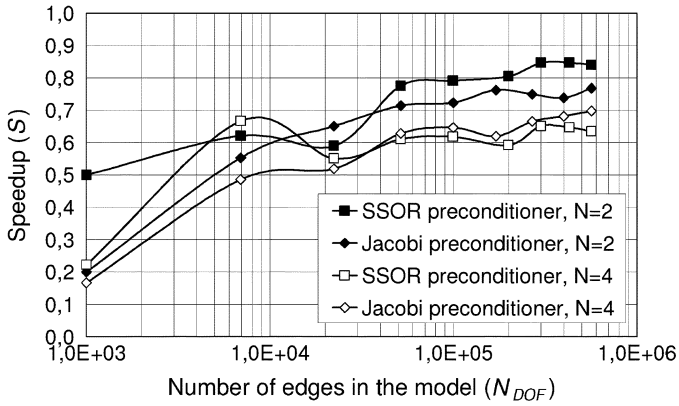

Fig. 4. Fixed speedup of the PCG algorithm versus number of edges in the FE model ($G = 1$).

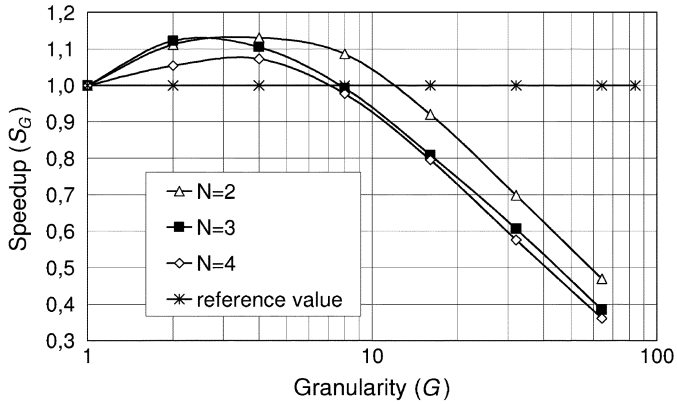

Fig. 5. Performance of the distributed SSOR preconditioner as a function of granularity ($N_{\mathrm{DOF}} = 99\,450$ edges).

$H(\mathrm{curl}, \Omega)$ tetrahedral edge elements [11]. The components of the $\mathbf{b}_n$ vector, according to time-domain multistep formulation, represents time dependent electromagnetic load.

The presented preconditioner is designed and developed to execute in a distributed memory environment. The communication between concurrent threads of the algorithm and the final form of parallelism of the code is controlled by the standard message passing interface calls [12]. The algorithm is validated on a cluster of workstations (COW) consisting of four processing nodes. Each computational unit is equipped with Intel Xeon 2.6-GHz CPU and 1-GB local RAM memory. The computers are connected through Gigabit Ethernet.

The performance of the presented SSOR distributed algorithm is compared to the point Jacobi preconditioner [5], [6]. The concurrent implementation of this algorithm requires a minimum number of data transfers. Therefore, the simple formulation of this method makes it useful as a relative benchmark measure in the distributed message passing environment.

IV. Comparative Studies

The general properties of the presented iterative solvers are defined by two metrics. The first one is the typical fixed speedup of the algorithm

$$S = \left.\frac{t_S(P=1)}{t_S(P)}\right|_{G=\mathrm{const}} \tag{2}$$

where t_s is the latency time of the PCG solver in a single step of the time-domain algorithm. The second coefficient describes relative speedup of the SSOR solver with different value of granularity

$$S_G = \left.\frac{t_S(G=1)}{t_S(G)}\right|_{N=\mathrm{const}}. \tag{3}$$

The different scale, typical test cases are used to verify the performance of the preconditioners. The largest benchmark model is approximated by $N_{\mathrm{DOF}} = 528\,660$ edges.

The fixed speedup of the solver for the coarse grained algorithm ($G = 1$) is presented in Fig. 4. The large-scale benchmark problems ($N_{\mathrm{DOF}} > 10^5$ edges) reveal the relative efficiency of the preconditioners in the COW system. The speedup of the PCG algorithm with both diagonal and SSOR preconditioners does not depend on the number of degrees of freedom (for the largest FE models), and it is below one. The size of transferred package depends on the size of benchmark problem, and it does not change the efficiency of the solver.

The PCG algorithm is executed repeatedly for each time step, and the initial guess distribution of the calculated field $\mathbf{e}_n$ is stated by the final solution of the foregoing step $\mathbf{e}_{n-1}$. The external electromagnetic excitation in the system is represented by smooth, physically constrained function, and the initial guess step in the $\mathbf{e}_n$ step is close to the final solution. In this case, the successive solution of the partial differential equation has near linear convergence is achieved, the number of iterations and number of data transfer commands are reduced.

The PCG algorithm with the SSOR preconditioner averagely consists of six–eight iterations. The PCG algorithm with Jacobi preconditioner needs about two times more iterations than the same algorithm with SSOR preconditioner. The diagonal preconditioner is relatively better parallelized, since there is no dependencies between concurrent subtasks. Experimental results prove, that there is no significant difference in the absolute time of computation. The gain in number of iterations in the SSOR preconditioner is leveled by some sequential and not-overlapped parts of the algorithm.

The curves of the S_G speedup for some medium and large-scale benchmark FE models are not some monotone functions (Figs. 5–7). The real speedup of the PCG solver is gained for the granularity number less than 10. The optimum value of the granularity factor G is approximately equal to 4. It depends on the number of processing workstations and the number of degrees of freedom in the model. The fine-grained formulation of the algorithm ($G > 10$) does not improve performance of the distributed computations.

There is no gain in the speedup of the presented algorithm for some small FE models. The curves of the S_G speedup are below reference level for the benchmark problems with the number of degrees of freedom (i.e., number of edges) less than 30 000 edges.

These effects arise from intensive communication and interdependencies between concurrent threads. The advantages of the distributed data processing are faded away. The faults of a loosely coupled multicomputer environment (i.e., synchronous

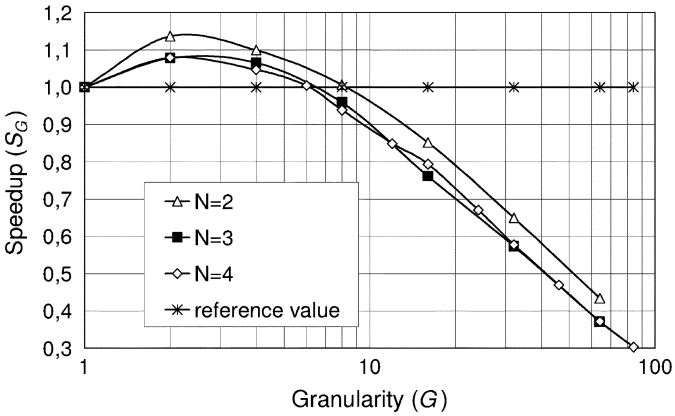

Fig. 6. Performance of the distributed SSOR preconditioner as a function of granularity ($N_{\mathrm{DOF}} = 528\,660$ edges).

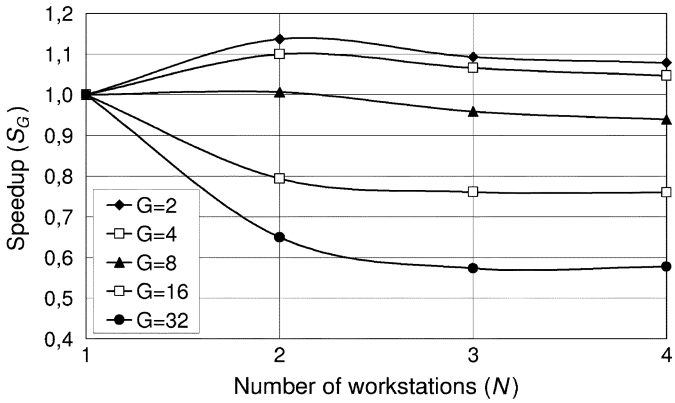

Fig. 7. S_G speedup of the PCG solver with SSOR preconditioner for different granularity coefficients ($N_{\mathrm{DOF}} = 528\,660$ edges).

data transfers, latency of data transfers, etc.) dominate over the profits of the algorithm.

V. Conclusion

The efficiency and flexibility of the SSOR preconditioner are shaped by the proper task decomposition. Particularly, the granularity of the task decomposition plays a significant role. The presented benchmark calculations show that the parallelized SSOR preconditioner gain the same performance as the most efficient, fully parallelized point Jacobi algorithm.

The profile of performance metrics of the SSOR preconditioner indicates its satisfactory scalability for the typical small COW. The computational speedup is highly determined by the number of broadcasted subresults and the bandwidth of the network.

The presented algorithm is not constrained by shape and order of implemented edge elements. The type of FE determines accuracy and stability of computations, convergence of the PCG algorithm, as well as performance of the presented distributed implementation of the SSOR preconditioner. This issue can be investigated in the future.

Acknowledgment

This work was supported in part by the Development and Scientific Found of Region Rhone-Alpes, France, under Grant 961 0 6550, and in part by project BTU W/WE/2/03.

References

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge, U.K.: Cambridge Univ. Press, 1994.

[2] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM, 2003.

[3] H. A. van der Vorst and T. F. Chan, "Linear system solvers: Sparse iterative methods," in *Parallel Numerical Algorithms*, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, Eds. Norwell, MA: Kluwer, 1997, pp. 91–118.

[4] H. A. van der Vorst, "High performance preconditioning," *SIAM J. Sci. Stat. Comput.*, vol. 10, pp. 1174–1185, 1989.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia, PA: SIAM, 1994.

[6] C. Vollaire and L. Nicolas, "Preconditioning techniques for the conjugate gradient solver on a parallel distributed memory computer," *IEEE Trans. Magn.*, vol. 34, no. 5, pp. 3347–3350, Sep. 1998.

[7] J.-F. Lee, R. Lee, and A. Cangellaris, "Time-domain finite-element methods," *IEEE Trans. Antennas Propagat.*, vol. 45, no. 3, pp. 430–441, Mar. 1997.

[8] B. Heise, "Nonlinear simulation of electromagnetic fields with domain decomposition methods on MIMD parallel computers," *J. Comput. Appl. Math.*, vol. 63, pp. 373–381, 1995.

[9] U. Navsariwala and S. Gedney, "An unconditionally stable parallel finite element time domain algorithm," presented at the Joint IEEE Antennas and Propagation Society/United Radio Science Institute (APS/URSI) Symp., Baltimore, MD, Jul. 1996.

[10] B. Butrylo, F. Musy, L. Nicolas, R. Perrussel, R. Scorretti, and C. Vollaire, "A survey of parallel solvers for the finite element method in computational electromagnetics," *COMPEL Int. J. Comput. Math. Elect. Electron. Eng.*, vol. 23, pp. 531–546, 2004.

[11] A. Ahagon, K. Fujiwara, and T. Nakata, "Comparison of various kinds of edge elements for electromagnetic field analysis," *IEEE Trans. Magn.*, vol. 32, no. 3, pp. 898–901, May 1996.

[12] K. Hwang and Z. Xu, *Scalable Parallel Computing Technology—Architecture—Programming*. New York: McGraw-Hill, 1998.

Manuscript received December 11, 2004.