Computing and Visualization in Science manuscript No.
(will be inserted by the editor)

A.V. Kononov · S.W. de Leeuw · C.D. Riyanti · C.W. Oosterlee · C. Vuik

Numerical performance of a parallel solution method for a heterogeneous 2D Helmholtz equation

the date of receipt and acceptance should be inserted later

Abstract The parallel performance of a numerical solution method for the scalar 2D Helmholtz equation written for inhomogeneous media is studied. The numerical solution is obtained by an iterative method applied to the preconditioned linear system which has been derived from a finite difference discretization. The preconditioner is approximately inverted using multigrid iterations. Parallel execution is implemented using the MPI library. Only a few iterations are required to solve numerically the so-called full Marmousi problem [12] for the high frequency range.

1 Introduction

The Helmholtz equation, which is also called a reduced wave equation, in scalar or vector form is often used to approximately model wave propagation in inhomogeneous media. The demand for reliable numerical solutions to such type of problems is repeatedly encountered in geophysical and optical applications [13], [14]. In geophysical applications, for example, wave propagation simulations are used for the development of acoustic imaging techniques for gaining knowledge about geophysical structures deep within the Earth's subsurface.

The discretization of the corresponding Helmholtz problem is usually based on finite difference (FD) or finite element discretization (FEM) schemes which are relatively simple and, at the same time, effective and increasingly popular. However, in order to maintain acceptable numerical accuracy in the FD or FEM solutions, fine enough grid spacings per wave length need to be employed [3], [4]. This implies that for most realistic cases the penalty in terms of computational costs and memory requirements is tending to be extremely high.

These severe limitations, as it will be shown further, may effectively be resolved by using the power of multiprocessor computer architectures, such as, for example, Linux computer clusters [15]. Unlike direct solution methods, iterative methods allow effective parallelization and require less memory utilization [7], and thus enable one to compute the solution of Helmholtz problems of practical size in reasonable time. In [8], for example, a parallel solver for the scalar Helmholtz equation is considered that is based on a parallel domain decomposition method. This solver is perfectly applicable to layered-like media, in which the layers are topologically similar to the layers of uniform thickness, and also the parameters vary smoothly within layers (this is usually the case in underwater acoustics). In [9], a parallel fictitious domain method has been used for the solution of 3D scattering problems such as a scattering of time-harmonic acoustic waves by an obstacle, showing good scalability properties.

The parallel approach, which is proposed in the present paper is rooted from the sequential code that is based on the method described in [1], [2]. So, the parallel algorithm is in general identical to the sequential one. This method appears to be advantageous because it is relatively simple and effective for the problems in which the media parameters can be strongly heterogeneous and the acoustic frequencies and the corresponding wavenumbers are relatively high (geophysical applications). Effectiveness of the algorithm is confirmed by the fact that only a few iterations are required to solve numerically the so-called full Marmousi problem [12] for the high frequency range. Moreover, the solver's relative simplicity is important for an optimal parallelization procedure.

A.V. Kononov · S.W. de Leeuw
Computational Physics Group,
PCMT, DelftChem
Delft University of Technology
E-mail: a.v.kononov@tudelft.nl

C.D. Riyanti · C.W. Oosterlee · C. Vuik
Numerical Analysis Group
Delft Institute of Applied Mathematics
Delft University of Technology

2 Model

Consider wave propagation in two space dimensions, which, in the frequency domain, for a constant density media, is described by the Helmholtz equation. The complex wave field amplitude due to external and internal sources obeys the fol-

lowing equation (here we follow [2])

$$\mathscr{A}u \equiv \left(-\partial_{xx} - \partial_{yy} - (1-i\alpha)\,k^2(x,y)\right)u = b(x,y), \qquad (1)$$
$$(x,y) : \Omega \in \mathbb{R}^2,$$

here $u = u(x,y)$ represents some physical quantity characterizing the field (i.e. pressure), $k(x,y) = 2\pi f/c(x,y)$ is the wavenumber, $c(x,y)$ the local speed of sound in a inhomogeneous medium, f is the frequency, $\alpha : (0 \le \alpha \ll 1)$ describes attenuation in the medium, and $b(x,y)$ is the source term. Boundary conditions at the boundary $\Gamma = \partial\Omega$ has to be chosen in such a way that an infinite space can be approximated by a bounded computational domain. In general, there are two possibilities of so-called absorbing boundary conditions [5] or as an alternative - the perfectly matched layer [6]. In the present case the second order Sommerfeld radiation boundary condition is used:

$$\mathscr{A}_\Gamma u = 0 : \quad \frac{\partial u}{\partial \nu} - ik\left(1 + \frac{1}{2k^2}\frac{\partial^2}{\partial \tau^2}\right)u = 0 \quad \text{on} \quad \Gamma, \quad (2)$$

where ν is the outward unit normal vector at the boundary, and τ is a vector pointing in the tangential direction. Numerical accuracy in the solution is controlled by the number of points per wavelength n_f , which is typically chosen to be $10-12$ points. Additionally, the number of wavelengths in a domain of size L equals Lf/c_{min}. Wavenumber k can be large, this means that the operator in Eq. (1) has both positive and negative eigenvalues, and, therefore matrix $\mathbb{A}$, the FD approximation of Eq. (1), is indefinite.

In geophysical applications, information on the local speed of sound in an inhomogeneous medium is usually stored in a large complex-valued array $\mathbb{C}[i,j]$. The so-called Marmousi synthetic data set is an remarkable example of such array, which was first released as a test for velocity estimation [12]. It is a complex acoustic 2-D data set based on the geology of the Cuanza basin in Angola, see Figure 3. Its structural style is dominated by growth faults which arise from salt creep and give rise to the complicated velocity structure in the upper part of the model. Later it was discovered that many numerical algorithms used for wave propagation modeling have failed to produce satisfactory results for the medium with such velocity profile. Therefore the Marmousi data set has become a popular very effective test for new numerical algorithms.

3 Numerical solution method

We consider the well-known 5-point discretization stencil with truncation error $O(h^2)$. In stencil notation it reads as:

$$A_h \approxeq \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 - h^2(1-i\alpha)k^2(x_i,y_j) & -1 \\ & -1 & \end{bmatrix} \qquad (3)$$

By applying this discretization to Eq. (1), (2) one obtains the following linear system

$$\mathbb{A}\boldsymbol{\phi} = \mathbf{b}, \quad \mathbb{A} \in \mathscr{C}^{N\times N}, \boldsymbol{\phi}, \mathbf{b} \in \mathscr{C}^N \qquad (4)$$

where N is the number of unknowns in the computational domain Ω_h. The sparse matrix $\mathbb{A}$ in Eq. (4) is complex valued due to both boundary conditions and the damping term in Eq. (1). Moreover $\mathbb{A}$ is in general symmetric, non-Hermitian, indefinite and, due to accuracy requirements, large for high wave-numbers and large computational domains. The numerical solution of system (4) is obtained by an iterative method applied to the preconditioned linear system, namely preconditioned Bi-CGSTAB [10] is used, which converges somewhat faster than other Krylov subspace methods (e.g. GMRES, QMR). The preconditioned system reads as

$$\mathbb{A}\mathbb{M}^{-1}\boldsymbol{\psi} = \mathbf{b}, \quad \text{with} \quad \boldsymbol{\phi} = \mathbb{M}^{-1}\boldsymbol{\psi}, \qquad (5)$$

where the preconditioner $\mathbb{M}$ proposed in [2] is, in fact, the damped version of the operator given by Eq. (1)

$$\mathscr{M} \equiv -\partial_{xx} - \partial_{yy} - (\beta_1 - i\beta_2)\,k^2(x,y), \qquad (6)$$

where $\beta_{1,2} \in \mathbb{R}$ are adjustable parameters. The method of choice is with the parameters is $(\beta_1, \beta_2) = (1, 0.5)$, as was determined in [2]. Boundary conditions for the preconditioner are identical to the original conditions (2). The preconditioner is approximately inverted by using multigrid one iteration [18].

The choice for multigrid as an inner solver is based on the study of a class of preconditioners for Helmholtz type problems, which was carried out in work [1].

Multigrid Components. Here we use standard multigrid coarsening that is doubling the mesh size h in every direction [18]. For smoothing the point-wise Jacobi relaxation with under-relaxation is used, which is well-parallelizable. The Galerkin coarse grid operator is used for the discrete coarse grid operators $M_{2h}, M_{4h}, \ldots$, which is defined as follows:

$$M_{2h} = R_h^{2h} M_h P_{2h}^h, \quad M_{4h} = R_{2h}^{4h} M_{2h} P_{4h}^{2h}, \quad \ldots, \qquad (7)$$

where M_h corresponds to discretization of Eq. (6) on the h, grid, R_h^{2h} and P_{2h}^h denote the restriction and prolongation operators, respectively. For heterogeneous problems, the Galerkin coarse grid discretization is a natural choice. Moreover, for the boundary conditions containing first and second derivatives, the Galerkin coarse grid discretization defines the appropriate coarse grid boundary conditions automatically.

The prolongation operator is based on operator-dependent interpolation, which is similar to de Zeeuw's transfer operators [11]. As the restriction operator the full weighting operator is employed. The choice for the combination of a full weighting restriction and the operator-dependent interpolation is based on the fact that it brings a robust convergence for a variety of Helmholtz problems with constant and non-constant coefficients, especially for the case of strongly varying coefficients, as in the Marmousi problem discussed further.

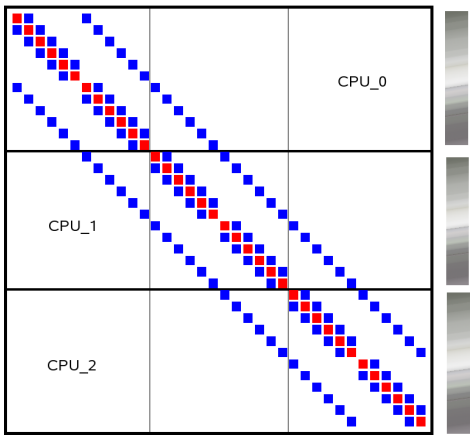


Fig. 1 Matrix $\mathbb{A}$ structure and data mapping onto 3 CPUs.

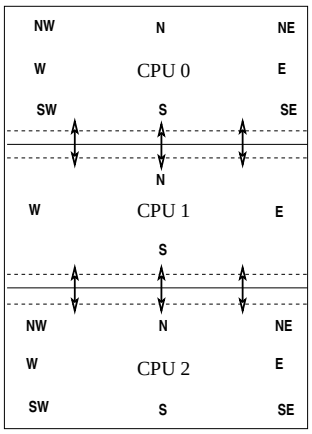


Fig. 2 It is convenient to introduce local coordinate system $\{N,S,E,W\}$, in which a particular processor receives data packets from North or (and) South.

4 Parallel program structure

As already indicated, using a parallel computing environment a program that is able to take advantage of such computing power is needed to solve realistic wave propagation problems in 2D and especially in 3D.

Let us consider the functional structure of the parallel program. Main functional blocks of the numerical solution procedure, namely the iterative solution block and the multigrid preconditioner, are inherently data parallel. Due to this, parallelization of the sequential program can essentially be based on the data parallel concept [17], [18]. Following this concept, the program data originated by the problem (the matrix, the solution vector and subsidiary storage vectors) has to be distributed between processors of a cluster. For the data exchange between processors, we use the well-known standard MPI library [16].

It is natural to decompose the matrix $\mathbb{A}$ and vector ϕ components as shown in Figure 1. Such a decomposition can be classified as rowwise block-striped decomposition. It has to be noted that the communication pattern (i.e. the amount and structure of the data) is completely determined by the type of decomposition. For our numerical test we use a rectangular computational domain, which is usually used in geophysics. In order to minimize the amount of data to be exchanged between computer nodes the computational domain has to be partitioned in the direction that is perpendicular to the longest dimension. The rowwise matrix decomposition corresponds to rectangular parallel strips which are partitioning the domain, see Figure 2.

It has to be noted that such decomposition is not quite optimal when the number of processors P is rather large $P \gtrsim \sqrt{N} \approx 10^2$. Therefore, it is quite pragmatic to assume that the dimension of the target problem is rather large and the number of processors available is considerably less than the problem size.

We now consider the main operations which are required by the program in some detail.

- *Matrix setup.*
- *Preconditioner setup.*
- *Matrix vector multiplications.*
- *Vector updates.*
- *Dot products.*
- *Preconditioning operations.*
- *Solution output.*

All these operations are performed in fully in parallel, except some preconditioning multigrid operations which are performed sequentially (for optimal performance) starting from a certain coarse level, which, in turn, is computed during the setup phase.

During *Matrix setup* the program has to read a large binary file that contains the data related to the local wave velocities. This operation has been rewritten in parallel mode by setting for each MPI-process a special data offset, so that each processor reads in parallel its own data chunk from the binary file which is usually stored at the so-called master node. Also at this stage, at each processor the descriptor structure is created, which describes the identification number of neighboring processor, the boundaries, i.e. if a particular boundary belongs to computational domain boundary or it is the data partition between processors. The *Preconditioner setup* consists of creating at each processor node the local coarse grid operator matrices together with the depended prolongation and restriction weights arrays. Also at this stage the number of the multigrid level (*last in parallel - LIP*) is computed until which multigrid computations are performed in parallel. The next coarser levels are computed in sequential mode without communication between MPI-processes. In turn, the LIP level is found if the number of grid points stored at the particular processor node is less than some predefined number (we use 100 points), which depends on hardware parameters and can be found from test runs.

The *Matrix vector multiplication* is one of the key operations used by the iterative solution procedure. It is supposed to be implemented with the use of the most effective parallel approach. Although at the finest level the 5-point stencil

is effectively used, at the coarse levels some zero stencil elements become non-zeros during the coarse grid operators setup. Therefore, it is convenient to store the matrix by 9-point finite difference stencils as a linear array, then matrix-vector multiplication has the following form

$$y_{ij} = \sum_{l=-1}^{1} \sum_{m=-1}^{1} A_{ij[lm]} x_{i+l\,j+m}, \tag{8}$$

here the summation indices l, m describe the summation over stencil components stored in the row related to a i, j grid point. From this equation it is seen that in order to accomplish matrix-vector multiplication the neighboring processors have to exchange the adjacent solution vector components. Such communication, for example, has been implemented by the MPI library functions [16]:

```
MPI_Send_Init(...)
MPI_Recv_Init(...)
MPI_Startall(...)
MPI_Waitall(...)
```

The *Vector updates* do not require any communication between neighboring processors, which means in particular that this operation scales perfectly.

The *Dot products* and similar operations effectively need one global communication communication between all MPI-processes, which can implemented using:

```
MPI_Allgather(...)
```

The *Preconditioning operations* consist of the prolongation and restriction operations, and Jacobi pre- and post-smoothing operations. As already mentioned, after some predefined multigrid coarse level the program execution switches from a parallel to a sequential mode in order to optimize performance. Such a program flow is governed by a special variable, which stores the current multigrid level and controls the behavior of all functional units of the parallel program.

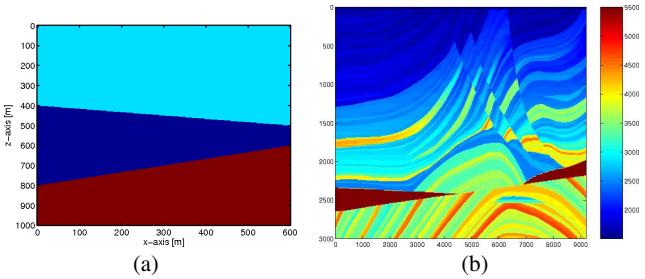
The *Solution output* is performed in a parallel regime. Each MPI-process outputs a locally stored part of the solution vector into its own file. Then, these output files can be merged into the global solution output file if necessary.

5 Parallel performance results

An effective way to assess the performance of a parallel program is to measure its execution time T as function of the number of processors $\mathscr{N}_p$. Additionally the so-called *parallel efficiency* can be evaluated that indicates how well a parallel program scales. It is defined as follows

$$E(\mathscr{N}_p) = \frac{1}{\mathscr{N}_p} \frac{T(1)}{T(\mathscr{N}_p)}. \tag{9}$$

Accordingly, if a program scales well then its efficiency is ≈ 1. Otherwise, low efficiency indicates that communication overhead prevails over computation. For the tests we have



(a) (b)

Fig. 3 *Models considered: (a) the wedge problem, (b) the Marmousi problem.*

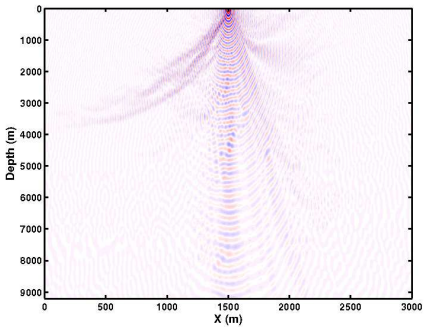


Fig. 4 An example of acoustic filed pattern for the case of the Marmousi problem.

Table 1 Wedge: f=60 Hz, Grid=481x801, Number of iterations = 32, Damping 5%

$\mathscr{N}_p$	**1**	**2**	**4**	**8**
Time	47.98	24.60	13.11	6.91
Mem.	286.5	164.2-165.6	92.8-108.9	54.7-69.9
Speedup	1	1.95	3.66	6.95
MF: Time	58.68	30.0	15.72	8.14
MF: Mem.	232.2	137.1-138.4	79.2-95.4	48-63.1
Speedup	1	1.95	3.73	7.2

chosen two model problems: the wedge problem and the Marmousi problem. The test examples differ substantially in the complexity of the sound velocity profile, see Figure 3, which, in turn, affects the number of iterations.

The wave field in a domain is excited by a harmonical delta-like source that is applied at the top of the computational domain. An example of field pattern for the case of the Marmousi problem is shown in Figure 4. We also compare a Full Matrix (FM) program which stores the coefficient matrix $\mathbb{A}$ with a so-called Matrix-free MF variant, which only stores the main diagonal (i.e. a variant with optimized memory usage). For the tests we use a Linux cluster which consists of a number single processor PC's, namely *AMD Athlon(TM) XP 2600+* interconnected with an *Ethernet* switch.

Results averaged over few runs are shown in Table 1. The table presents CPU-time, memory utilization, and relative speedup for the wedge problem and the source frequency $f = 60$ Hz, grid size 481x801, and convergence achieved after 32 iterations. Here execution time denoted by 'Time' is

Table 2 The Marmousi: f=30 Hz, Grid=2001x534, Number of iterations = 38, Damping 5%

$\mathcal{N}_p$	**1**	**2**	**4**	**8**
Time	158.01	79.76	40.90	21.59
Mem.	793.7	426.3-428.4	225.3-246.3	126.2-147.2
Speedup	1	1.98	3.86	7.32
MF: Time	139.64	69.92	35.48	18.28
MF: Mem.	643.4	351.2-353.2	191.9-212.9	109.5-130.5
Speedup	1	1.99	3.93	7.63

Table 3 The Marmousi: f=60 Hz, Grid=2501x751, Number of iterations = 75, Damping 5%

$\mathcal{N}_p$	**2**	**4**	**8**	**10**
Time	223.0	110.06	59.3	49.01
Mem.	530.-532.	270.-274.2	138.0-141.1	112.2-115.05
Speedup	1	2.03	3.76	4.55

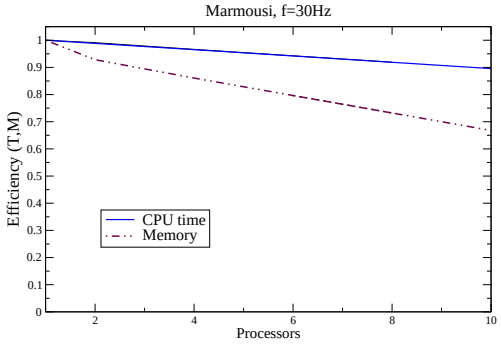


Fig. 5 Parallel efficiency of the MF program.

measured in seconds, and memory usage per processor node denoted by 'Mem.' is measured in MBytes. Data shown in Table 1 indicates that for a relatively small problem the FM program is faster than the MF program because of recomputing of the matrix elements at the boundary of the domain at each iteration step. However, the MF program scales better than the FM program because of the better ratio between computation and communication. Moreover, it is clear that the MF program uses less memory and it will gain more for really large problems. In Table 2 the data related to the Marmousi model are shown. Now the MF program outperforms the FM variant since the amount of inner (not related to the boundary points) computations prevails and extra time is necessary to invoke matrix elements from a relatively large array. Table 3 shows run data obtained using a larger grid for the MF program only because of its superior performance. In this test due to memory limitations, it was only possible to start the test on a two node computer configuration. Results show a very satisfactory scalabity of the program with respect to the wall clock time and memory usage. In Figure 5 the graphs of the parallel efficiencies for the full Marmousi case f=30 Hz are shown. Here, similarly to the parallel efficiency which is evaluated according to Eq. (9), we have evaluated the *parallel memory usage efficiency*. Moreover, it can be additionally noted that the nonzero small damping may substantially reduce the number of iterations. From the graphs the following conclusions can be drawn. The parallel scalability of the program is quite good. This is mostly due to the well parallelizable multigrid preconditioner that consumes most of the computational time. Moreover, better results can be achieved by using a faster network. At first glance one may notice that the memory usage does not scale well, however, one should bear in mind that a substantial amount of memory is allocated by the MPI library buffers. We have not run the program on larger numbers of processors, since our actual target is 3D. Having the 3D version ready, it is possible then to make further parallel optimization in order to achieve maximum parallel performance.

6 Conclusions

In this paper we have studied the numerical performance of parallel solution of a heterogeneous 2D Helmholtz equation. The most important conclusions of the numerical tests are the following:

- Memory and performance limitations for large problems can effectively be resolved by using a parallel computing approach.
- The multigrid preconditioner shows a satisfactory parallel efficiency.
- In the case of 3D problems even better parallel scalability may be expected.
- Using a parallel computing environment, the full Marmousi can be solved in only a few iterations, in reasonable time.

Acknowledgements The authors would like to acknowledge that the current research is financially supported by the Dutch Ministry of Economic Affairs under the Project BTS01044.

References

1. Erlangga, Y.A., Vuik, C., Oosterlee, C.W.: On a class of preconditioners for the Helmholtz equation. *Applied Num. Math.*, **50**, 409–425 (2004).
2. Erlangga, Y.A., Vuik, C., Oosterlee, C.W.: A novel multigrid preconditioner for heterogeneous Helmholtz problems. *to appear in SIAM J. Sci. Comp.* (2005).
3. Marfurt, K.J.: Accuracy of finite-difference and finite element modeling of the scalar and elastic wave equations.*Geophysics*, **49:5**, 533–549 (1984).
4. Bayliss, A., Goldstein, C.I., Turkel, E.: On accuracy conditions for the numerical computation of waves.*J. Comput. Phys.* **49**, 394–404 (1985).
5. Bamberger, A., Joly., P., Roberts, J.E.: Second-order absorbing boundary conditions for the wave equation: A solution for the corner problem. *SIAM J. Numer. Anal.*, **27**, 323–352 (1984).
6. Turkel, E., Yefet, A.: Absorbing PML boundary layers for wave-like equations. *Applied Num. Math.*, **27**, 533–557 (1998).
7. Berglund, G.Z.M., de Leeuw, S.W.: A feasibility study of the use of two parallel sparse direct solvers for Helmholtz equation on Linux clusters. *Reports of the Department of Applied Sciences.*, REPORT 03-01, Delft University of Technology (2004).
8. Larsson, E., Holmgren, S.: A parallel domain decomposition method for the Helmholtz equation. TECHNICAL REPORT 2000-006, Uppsala University, Sweden (2000).

9. Heikkola, E., Rossi, T., Toivanen J.: A parallel fictitious domain method for the three-dimensional Helmholtz equation. *SIAM J. Sci. Comput.***24:5**, 1567–1568 (2003).
10. Van der Vorst, H.A.: BI-CGSTAB: a fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.***13**, 631–644 (1992).
11. de Zeeuw, P.M.: Matrix-dependent prolongation and restrictions in a blackbox multigrid solver. *J. Comp. Appl. Math.*, **33**, 1–27 (1998).
12. Bourgeois A., Bourget M., Lailly P., Poulet M., Ricarte P., Versteeg R.: Marmousi, model and data, in Versteeg, R., and Grau, G., Eds., The Marmousi experience, *Proceedings of the 1990 EAEG workshop on Practical Aspects of Seismic Data Inversion: Eur. Assoc. Expl. Geophys.*, 5-16 (1991).
13. Bleistein, N., Cohen, J., Stockwell, J.Jr.: Mathematics of Multidimensional Seismic Imaging, Migration, and Inversion. Springer Verlag, Berlin (2001).
14. Peterson, A., Ray, S., Mittra, R.: Computational Methods for Electromagnetics. IEEE Press, New York (1998).
15. Bookman, C.: Linux Clustering: Building and Maintaining Linux Clusters. Sams Publishing, New York (2002).
16. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J.: MPI-The Complete Reference: Volume 1, The MPI Core. The MIT Press, London (1998)
17. Quinn, M.J.: Parallel Computing: Theory and Practice. McGraw-Hill, New York (1994).
18. Trottenberg, U., Oosterlee C. and Schüller, A.: Multigrid. Academic Press, London (2001).